2006

Acoustic classification using independent component analysis

James Brock

Acoustic Classification Using Independent Component Analysis

by

James L. Brock

A Thesis Submitted in Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Supervised by

Director, Laboratory of Intelligent Systems Dr. Roger Gaborski
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York
May 2006

Approved By:

Dr. Roger Gaborski
Director, Laboratory of Intelligent Systems
Primary Advisor

Carl Reynolds
Visiting Professor

Peter Anderson
Professor Emeritus

Thesis Release Permission Form

Rochester Institute of Technology

Department of Computer Science

Title: Acoustic Classification Using Independent Component Analysis

I, James L. Brock, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

James L. Brock

Date

Acknowledgments

I would first like to thank my thesis advisory committee, Dr. Roger Gaborski, Dr. Carl Reynolds, and Dr. Peter Anderson. I would also like to thank everyone that helped and supported me along the way to completing this work. Last, but not least I would like to thank Lemmy and God, even though they're the same person.

Abstract

This thesis research investigates and demonstrates the feasibility of performing computationally efficient, high-dimensional acoustic classification using Mel-frequency cepstral coefficients and independent component analysis for temporal feature extraction. A process was developed to calculate Mel-frequency cepstral coefficients from samples of acoustic data grouped by either musical genre or spoken world language. Then independent component analysis was employed to extract the higher level temporal features of the coefficients in each class. These sets of unique independent features represent themes, or patterns, over time that are the underlying signals in that class of acoustic data. The results obtained from this process clearly show that the uniqueness of the independent components for each class of acoustic information are legitimate grounds for separation and classification of the data.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Advancements in networking technology and globalization have spawned a quickly expanding digital media market. This market has given all classes of individual access to on-demand videos, music, movies, etc. and created the growing problem of how to identify, classify, organize, and manage this media. Traditionally, this problem has been addressed with humans classifying the media and constructing meta data stored with each file to identify its characteristics, because content recognition is a seemingly trivial task for the average person. It has become quite obvious in recent years that the amount of digital media present in the world is far too great to use this method to identify the characteristics of a file's content in an efficient manner.

In the music world specifically this problem has come to the forefront with the inception of online stores for digital music (iTunes, Yahoo! Music, RealMusic, etc.) and streaming audio services such as Pandora (based off of the Music Genome Project), Yahoo! Launch, and internet radio. These services use advanced queries of the user-specified meta data to search through their databases and cater to the listener's preferences. However, not only is musical preference extremely unique, but the classification of music is a matter of personal opinion as well, since there is no standard for musical genres and sub-genres. A song one person considers to be *classic rock* may be *oldies* to another and still *acid rock* to another, and so on. There is a clear and present need for a means by which to automatically, and

intelligently classify digital music, and other media, based on some loose criteria.

This problem can easily be extended to all kinds of acoustic information and databases. Records of speech, animal noises, and other sounds used in scientific research and surveillance are difficult to maintain without some working knowledge of the content of each piece of data. A general process for acoustic classification would enable the majority of this data to be automatically classified and produce meta data that would ease the cost of time and computing resources necessary to do so through traditional means. The feasibility of such a process is the primary motivation for this thesis research into a successful means by which all acoustic information can be automatically classified, beginning with music and extending it to world languages.

1.2 Auditory Cognition

The human auditory system has been the subject of research for quite some time and resulted in information concerning how to represent and mimic its basic behavior. It's been a consistent problem, however, to model the higher-level functionality and acoustic interpretation of human hearing and cognition. Though there is still a great deal of debate on which is best, most current models break down human hearing into hierarchal, functional layers much like the modeling of the visual cortex. Since much more work has been done in the field of computer vision and modeling the visual cortex, a great deal of the research done on the auditory cortex is based off of successful algorithms and concepts from the area of computer vision. One of these areas of research is the human ability to distinguish accurately between different types of sounds almost instantly. Human beings are able to distinguish the sources, characteristics, and general location of most sounds easily allowing them to classify them as significant or group them together. It is a current area of research developing algorithms that achieve this level of cognitive performance in response to acoustic stimulus.

This thesis will discuss and develop an approach to modeling the higher-level cognition of the human auditory system in a way that makes automatic classification of acoustic information using independent component analysis quit feasible. An audio processing algorithm will be constructed using Mel-frequency cepstral coefficients, principal component analysis, and independent component analysis to separate acoustic information in the form of music and foreign languages.

1.3 Thesis Outline

This thesis will begin with a background explanation of the major concepts and subject matter used in this work, including Mel-frequency cepstral coefficient, principal component analysis, and independent component analysis. Material and results from work done within the domain of signal processing, source separation, and acoustic understanding will be presented in support of this research. This chapter will also explain the biological inspiration for using the aforementioned techniques in modeling the human auditory system and its cognitive properties. The third chapter will explain the process of developing a novel process for acoustic information separation based on previous work and techniques used in this area. Model diagrams will demonstrate the flow and processes used to extract significant features, separate the feature set using independent component analysis, and then demonstrate the independence of groups of acoustic information. The process will also be explained conceptually to give a general overview of how this is able to support simple high-level acoustic classification while taking particular time to highlight the most significant parts of the algorithm.

Chapter four of this thesis will delve deeply into the technical aspects of the developed process, siting specific values, parameters, algorithms, and audio signal processing steps taken to achieve optimal results. Each step of the process will be explained in the context of the conceptual description and each modification and technical choice explained fully, as well as the effect of changing some variables and aspects of the design that were learned

along the way. This chapter will also explain how the process was organized and give details about the acoustic information and databases used for this research.

The fifth chapter will explain the results achieved and evaluate the algorithm's performance, making note of where, how, and why the process either succeeded or gave marginal results. The results will be evaluated and explained in order to correlate unexpected performance with the particular part of the process that is the source of any discrepancies. Finally, the sixth chapter will give final conclusions about the thesis work and results achieved in a more general context, also suggesting areas for future research to be done in this problem domain.

Chapter 2

Background

2.1 Biological Inspiration

The human auditory cortex is a complex cognitive system, able to make extremely selective and complex decisions in an instant. This performance has long been desired in the field of intelligent systems and signal processing. Research is always being done to try to model, mimic, and further understand this system. There are many models for how different researchers believe the auditory cortex functions at the neurological level. One common model is shown in figure 2.1. Where the objects labelled as PFC, PP, and PB are the prefrontal connections, posterior parietal cortex, and parabelt cortex respectively. These are the biological parts of the brain that handle the highest levels of auditory interpretation and interfacing with the rest of the brain [23]. The exact function and mechanics are beyond the scope of this work. The other regions depicted in the diagram, however, are the exact part of the auditory cortex that this approach to acoustic classification is trying to model and will be explained further.

The primary purpose of this diagram is to map the ”Where” and ”What” paths of the auditory cortex, with ”Where” shown in Red and ”What” shown in Green [23]. The ”Where” path is the path through the brain audio signals take to determine the spatial information of the sound. The ”Where” path is the same thing, but for identifying the type of sound. The set of rectangular areas labeled *Cortex* and *Thalamus* on the right side of the diagram depict how the intermediate and high-level functionality of the auditory cortex is organized.

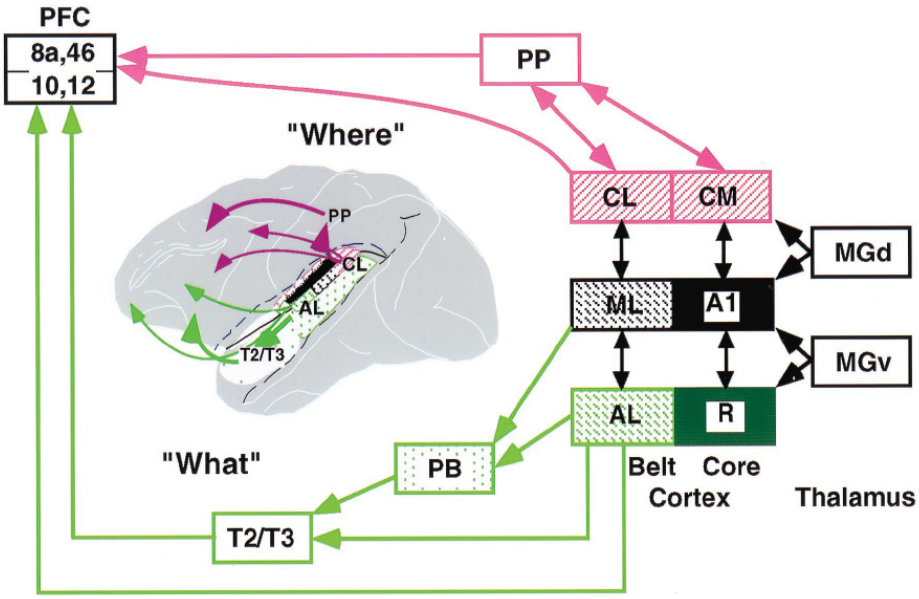

Figure 2.1: Auditory Cortex Model[2]

Those regions are organized into three subsections, the caudal belt (labeled *belt* in figure 2.1), the core, and the thalamus [23]. The thalamus is the first step in higher-level auditory cognition, as acoustic information from the ear enters the auditory cortex via the medial geniculate nucleus (MGN, sometimes combined with the cochlear nuclei) within it, making it the primary auditory relay nucleus. The MGN is broken up into ventral and dorsal parts, with the dorsal part focused on spatial information, and the ventral part focused on temporal and frequency information.

As shown in figure 2.1, audio data flows from the two parts of the MGN to the two parallel spatial cortex regions (CL and CM) and identifying cortex regions (R and AL). The arrows depicting the flow of information have been biologically proven by tracing connections between the regions in various human and non-human primates [23]. There is also, a couple unifying regions of the cortex (ML and A1) that respond to both types of audio data, but perform some distinction to pass appropriate information onto the other regions. None of the regions' functions are exact, all three caudal belt regions respond to some degree to

spatial information, and ML and A1 both respond well to all kinds of acoustic information. However, the important aspect of the auditory cortex's functionality to take away from this is that both spatial information and other information, such as temporal features, frequency changes, and amplitude changes, are processed in parallel and the human brain does make a distinction between them [23]. The purpose of the work in this thesis is to model the higher-level cognition of the "What" path, using identifying temporal themes with regards to frequency to show that somewhat intelligent distinctions can be made computationally.

For relevant audio processing applications, researchers are continually trying to model this higher-level operation quantitatively. Recent work has determined that independent component analysis (ICA) is a sufficient method for modeling some of the higher-level interpretation of sensory information in humans [16],[17]. Many intelligent systems applications using independent component analysis exist in the area of computer vision. Specifically, it has been shown that music files processed with principal component analysis exhibit statistical independence that could possibly be used to develop unique representations of particular genres, and proposed that a better separation of classes of sound could be achieved with independent component analysis[16].

These cognitive component analysis techniques employ the use of some preprocessing functions to produce a representation of auditory information similar to what is found at the low to middle levels of the primary auditory cortex. Mel-frequency cepstral coefficients (MFCC's) have been shown to mimic the way the human ear responds to sound, with a logarithmically higher response to lower frequency ranges [18]. The human cochlea, located in the inner ear and pictured in figure 2.2, is able to react to sounds more acutely at lower frequencies because they are interpreted first in that part of the ear.

With an accurate model of the low-level interpretation of acoustic information in humans, algorithms can be derived that estimate the cognitive functionality of the higher-level cortex layers, creating a biologically inspired system able to interpret sound in some of the same ways humans do. It is therefore theoretically viable, and has been suggested as an

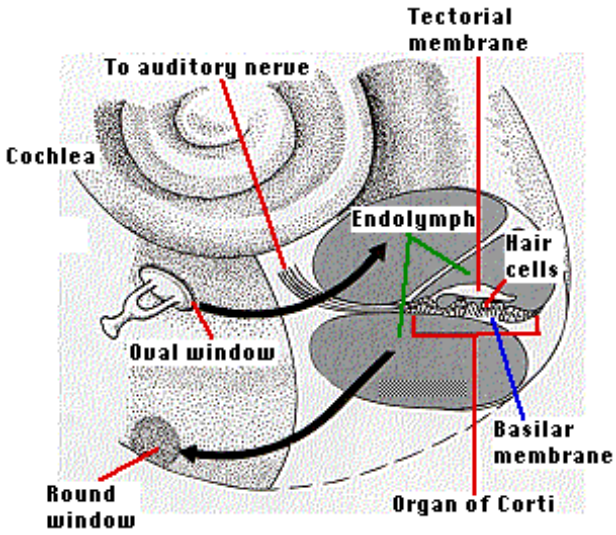

Figure 2.2: Human cochlea, which reacts in different areas to different frequencies of sound [2]

area of further research in recent works, that the use of both Mel-frequency cepstral coefficients and independent component analysis can potentially produce an acceptable model of the upper levels of the human auditory system that can perform the general separation of acoustic information[16]. The remainder of this chapter will explain these techniques and how they will be applied to the development of a process that could be used to classify acoustic information with reasonable success.

2.2 Mel-Frequency Cepstral Coefficients

Mel-frequency cepstral coefficients are a popular method for low-level feature extraction in audio processing [18]. The coefficients comprise a good representation of the dominant features in acoustic information for a given window of time. The process for developing the signal's frequency response and the coefficients is depicted in figure 2.3 along with the Mel-scale filter bank.

The first step is to break the input signal down into windows of time which will usually

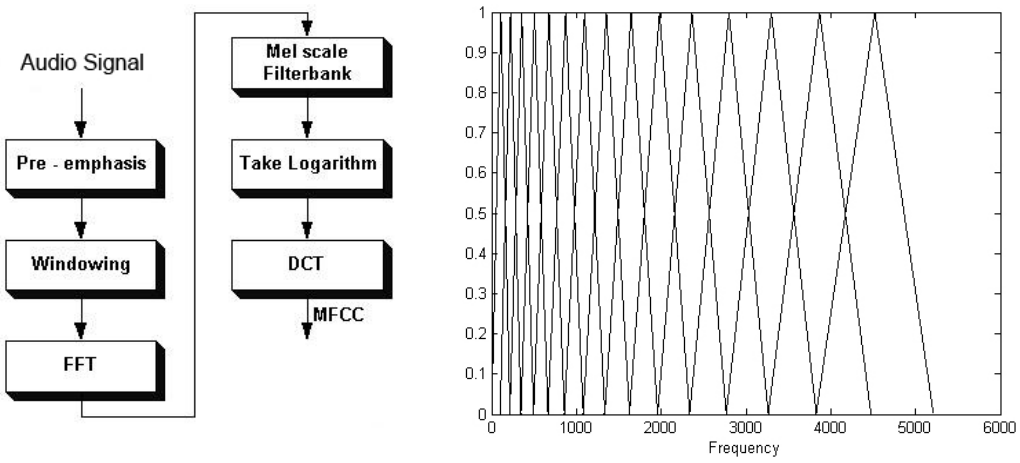

Figure 2.3: MFCC process(left)[5] and Mel-scale filter bank(right)[22]

have some overlap in order to comprehensively capture the signal's temporal features and changes. A fast Fourier transform (FFT) is performed on the windowed signal. The FFT is an efficient, and thus faster, algorithm for computing the discrete Fourier transform (DFT) of a particular signal and its inverse. A discrete Fourier transform essentially just separates out the component frequencies from a signal, based on the theory that any real signal is a linear combination of a number of sine waves at different frequencies. In this transform, the sequence of N complex numbers $x_0, x_1, ..., x_{N-1}$ are transformed into the sequence of N complex numbers $X_0, X_1, ..., X_{N-1}$ by equation 2.1[4].

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk}; \qquad k = 0, 1, ...N-1; \tag{2.1}$$

The output from this algorithm is a breakdown of the frequency components of the original signal, which as stated above can be linearly combined to form the original signal. These frequency separated signals are then passed through the set of bandpass filters found in the Mel-frequency scale. The Mel-scale is organized logarithmically (see figure 2.3) to better represent the actual auditory response of the human ear, with a more defined frequency response at the lower frequencies audible to humans. The number of bandpass

filters used corresponds to the number of coefficients (typically 11-13, but can be as many as 30[11]) generated for each window of time. The result of sending the frequency response data through the Mel-scale bandpass filters is a set of intermediate signals that represent the frequency response of the original signal within the bandpass filter ranges. This generalized frequency response is one of the two significant pieces of information that can be retrieved from the MFCC process. The response signals are a linear combination of the original signal, an example is given in figure 2.4. In many speech processing algorithms, these responses are used to reconstruct the source signal.

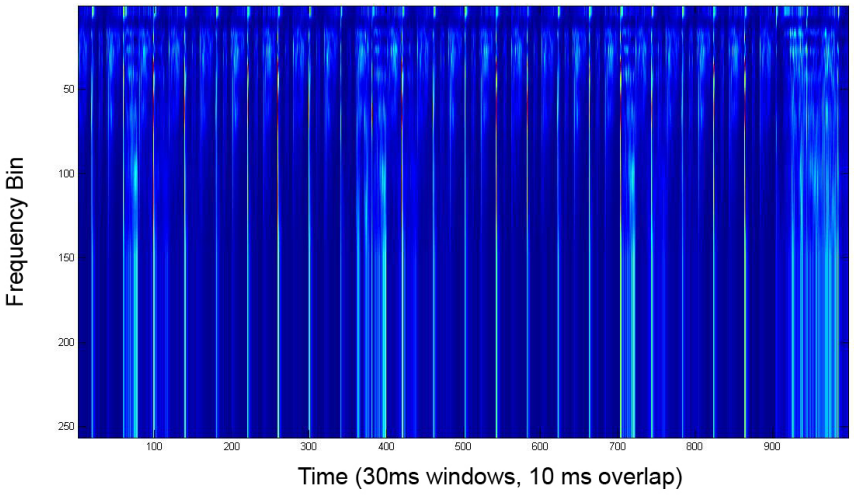

Figure 2.4: Band-passed frequency response of a jazz song

The second piece of information of interest from this whole process is the Mel-frequency cepstral coefficients themselves. The coefficients are generated by taking the log of the band passed frequency response and calculating the discrete cosine transform (DCT) of each intermediate signal. Typically, the Mel-frequency cepstral coefficients (C_i) of a frame of audio data are the output from the DCT transform, given by equation 2.2 [7]

$$C_i = \sum_{j=1}^{M} m_j Cos\left[\frac{i\pi}{M}\left(j - 0.5\right)\right]; \quad m_j = log_e\left(Y_j\right); \tag{2.2}$$

$$i = 0, 1, 2, ...N; \quad N < M;$$

where Y_j is the output magnitude of the j^{th} Mel-scale filtered signal and M is the total number of Mel-scale filters in the filter bank analysis [7]. The resulting coefficients represent the response of each particular frequency band at a particular window of time in the audio sample. Therefore, for any window of time the relative response of the range of frequencies audible to humans is given, with larger coefficients corresponding to a strong feature for the window. The matrix output by determining the MFCC's for an audio signal is therefore accepted as a good statistical depiction of the original signal's significant features in the frequency domain. This technique has been widely used for feature extraction in speech recognition as well as in applications processing digital music[18], [7], [9].

2.3 Principal Component Analysis

2.3.1 Theoretical Background

Principal component analysis (PCA) is a statistical transform for multidimensional data that represents relevant patterns in data and outputs the information in a way that emphasizes their similarities and differences [26]. Principal component analysis is used for reducing dimensionality, identifying significant patterns, and has been used extensively in image and audio processing [16] [26]. PCA does this by accounting for as much variance in the data set as possible with each principal component. The process for computing the principal components of a set of multidimensional data is based on the common statistical concepts of variance, covariance and eigenvectors.

For a set of one dimensional data (dimensions can also be thought of as variables) the spread, or range, of the information is commonly represented by standard deviation and variance. Variance (s^2) is simply the standard deviation of the data squared and represented by equation 2.3.

$$s^2 = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2}{(n-1)} \tag{2.3}$$

Because standard deviation and variance only operate on 1-dimensional data, a measure is needed for multidimensional data sets; covariance is such a measure. It represents the variance of dimensions of data with respect to each other. Therefore, the covariance of two sets of observed data will determine how much the observations of one dimension change with a given change in the other dimensions data. The equation for covariance of two dimensions of data X and Y, with n observations each is given in equation 2.4.

$$cov\left(X, Y\right) = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{(n-1)} \tag{2.4}$$

Because covariance represents a change in one dimension relative to a change in another, then a positive covariance value indicates that as one set of data increases, the other set of data will increase as well. Inversely, a negative covariance value indicates that as one dimension increases the other *decreases*. Thus, a covariance value of zero means that the two dimensions' changes are statistically independent of each other and do not relate linearly. This relationship can be evaluated for any number of dimensions, with each dimension of data being separately related to every other dimension. The information derived from doing this is commonly organized into a covariance matrix containing the covariances of all possible dimensional relationships. A covariance matrix for a set of data with three dimensions x, y, and z is shown in equation 2.5 [26].

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix} \tag{2.5}$$

Down the main diagonal, you can see that the covariance value is between one of the dimensions and itself, which is just the variance for that dimension. The other noticeable

point is that since $cov(a, b) = cov(b, a)$, the matrix is symmetrical about the main diagonal [26]. Calculating a covariance matrix for a high-dimensional set of data is useful, but difficult to interpret. Principle component analysis allows for dimension reduction and interpretation of higher-level patterns and relationships in the data that are not immediately evident from the information contained in the covariance matrix. Yet, the covariance matrix of relationships between each observed dimensions' data is crucial to determining the principal components of a set of data, which is completed with the calculation of the eigenvectors and eigenvalues of the covariance matrix. Eigenvectors and eigenvalues can be calculated by a number of different methods that solve the matrix algebra equation 2.6

$$C \cdot V = V \cdot D \tag{2.6}$$

where D is the diagonal eigenvalue matrix and V is the associated eigenvector matrix of the covariance matrix C [4]. Eigenvectors, also referred to as principal components, latent dimensions, or principal features, represent the characteristic themes in a multi-dimensional dataset. These eigenvectors will account for as much of the variance in the original data set as possible. In fact, the German term *eigen* can be loosely translated into "peculiar" or "characteristic" [4]. Each eigenvector will have with it one associated eigenvalue in the diagonal matrix, where the eigenvector with the highest eigenvalue is the n-length vector of an n by n covariance matrix that represents the greatest amount of variance in the entire data set (not just one of the original dimensions). Each subsequent eigenvector with a gradually smaller eigenvalue, therefore, represents a slightly smaller degree of variance in the data and is orthogonal to all other eigenvectors. Thus the eigenvectors of the covariance matrix, sorted by their eigenvalues, are the features or patterns within the original data sorted by significance. Typically this information is returned by a principal component analysis algorithm pre-sorted.

2.3.2 Process

The exact process for producing principal components is relatively straight forward, beginning with subtracting the mean from each dimension of data for normalization. Then, through whatever means is most convenient, calculate the covariance matrix for all of the dimensions of the data. The last two steps are the evaluate the eigenvectors and eigenvalues of that covariance matrix, and simply sort the eigenvectors based on their associated eigenvalues. With most modern statistical processing software this can be accomplished in just a few function calls, or even just one. Because each eigenvector produced is orthogonal across all dimensions to the other principal components produced, the first principal component identifies the vector that will account for the most variance in the data, and the second component accounting for the most variance in the data in the second orthogonal dimension. This is best visualized when a set of 2-dimensional data is plotted with it's calculated eigenvectors drawn over it as in figure 2.5. It is clear from the plots that the eigenvectors represent the variance of the data, but not along the traditional x and y axes. Instead the vectors can also be used as axes to reorganize the data.

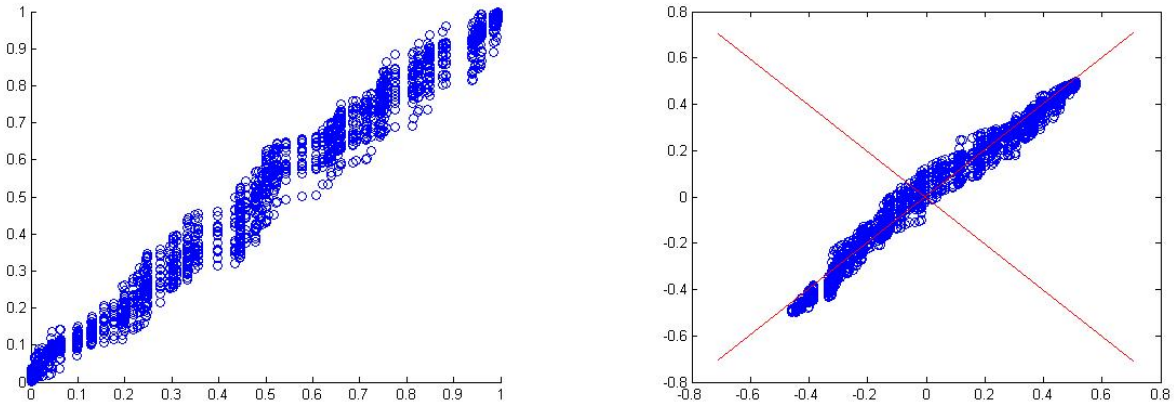

Figure 2.5: Raw data(left) and normalized data with eigenvectors superimposed(right)

One of the primary reasons for using an algorithm like PCA is to reduce dimensionality. This is done by only using the m ($m < n$) eigenvectors with the highest eigenvalues. At

the very least, eigenvectors with an eigenvalue of zero are always considered insignificant, and can be thrown out as they account for none of the variance in the data set. The retained eigenvectors will represent the most significant patterns and features of the data with minimal loss of information, but the amount of data has now been reduced making any further calculations simpler and less computationally expensive.

2.3.3 Examples

There is a huge number of areas of research using principal component analysis, such as economics, sociology, physics, and intelligent computing. They are particularly popular in signal processing with images, video, and audio [26],[27], [25]. One of the most successful applications of PCA in signal processing, has been the use of eigenfaces for identification, recognition, and classification of humans in surveillance applications. This section will describe the use of PCA for generating eigenfaces because it not only demonstrates the significance of the process, but is a good visual representation of how PCA can represent patterns and themes in data with a large number of dimensions.

The most effective algorithms for face detection and recognition have been the ones that most closely model the actual performance of the human visual system's ability to quickly and accurately discern learned images. The use of a small set of representative images, called eigenfaces, characterizes the principal components of a set of known faces. The principal components of the known images represent a biologically inspired way of representing the meaningful features of the set of pictures. These eigenfaces are also used to reduce dimensionality of the data set to only include the most meaningful features of the image set.

The process for generating eigenfaces is quite simple, beginning with a set of face images of some uniform size (m by n). The images are individually gray scaled and then vectorized. The matrix of these image vectors (1 image per row) is considered the dataset for principal component analysis. The images will each represent one dimension of mn observations. This matrix is then passed through PCA, and the resulting eigenvectors will

each display an eigenface when converted from their mn-length eigenvector to an m by n matrix as seen in figure 2.6.

Figure 2.6: Example eigenfaces

As you can see from the output images, there are clearly ghost-like facial features that represent the variance in the facial features of all of the input images. The dimensionality of the image information can be reduced by throwing out the eigenfaces with the lowest eigenvalues because they will not represent a large amount of the original face data. In the case of eigenfaces, each original image can still be almost entirely reconstructed through some linear combination of the eigenfaces that remain. The coefficients of that linear combination are the important information for this recognition technique. Figure 2.7 depicts the coefficients being used to reconstruct one of the original images.

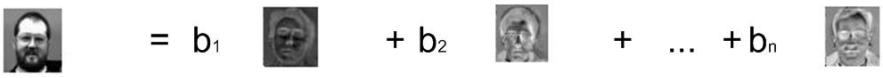

Figure 2.7: Linear combinations of eigenfaces reconstruct original images

When new images are introduced to the system trained with these eigenfaces and coefficients, the new images are projected onto the face space to determine their set of coefficients

relative to the known eigenfaces. The coefficients are then compared to all the known sets of coefficients to determine which known face matches the new face. This is a very basic form of using eigenfaces for human face recognition and classification, and it has come into widespread use in recent years. This is significant proof of the high-level, cognitive processes that statistical transforms like this are capable of[27].

2.4 Independent Component Analysis

2.4.1 Introduction to ICA

Independent component analysis belongs to a class of blind source separation (BSS) methods for separating data into underlying components, where such data can take the form of images, sounds, marketing data, or or stock market prices. ICA is based on the simple, and physically realistic assumption that if different signals are from different physical processes, then those signals are statistically independent[1]. Independent component analysis takes advantage of the inverse of this assumption, which leads to a new assumption that is logically un-sound but which works in practice. That assumption is that if statistically independent signals can be extracted from signal mixtures then these extracted signals must be from different physical processes[1]. ICA has been applied to problems in fields as diverse as speech processing, brian imaging, audio and visual recognition, and stock market prediction. Currently, research is being done with independent component analysis in many different fields, and with its success in speech processing it is reasonable to say that it has a large potential for more general acoustic separation and classification.

Independent component analysis is of interest to a wide variety of scientists and engineers because it promises to reveal the driving forces behind a set of observed data signals, including neurons, cell phone signals, stock prices, and voices[1]. The most common example of using independent component analysis for blind source separation is the *cocktail party problem*, where there are n people speaking at a party that is being recorded by n microphones somewhere in the room in the ideal case. A good blind source separation

algorithm, such as ICA, will be able to take the n mixtures of voices from the microphones and separate out the individual voice signals from them. A depiction of how this problem works can be found in figure 2.8. The microphone signals shown contain information from each of the original input signals, which means that all of the different physical processes contribute to all of the mixtures.

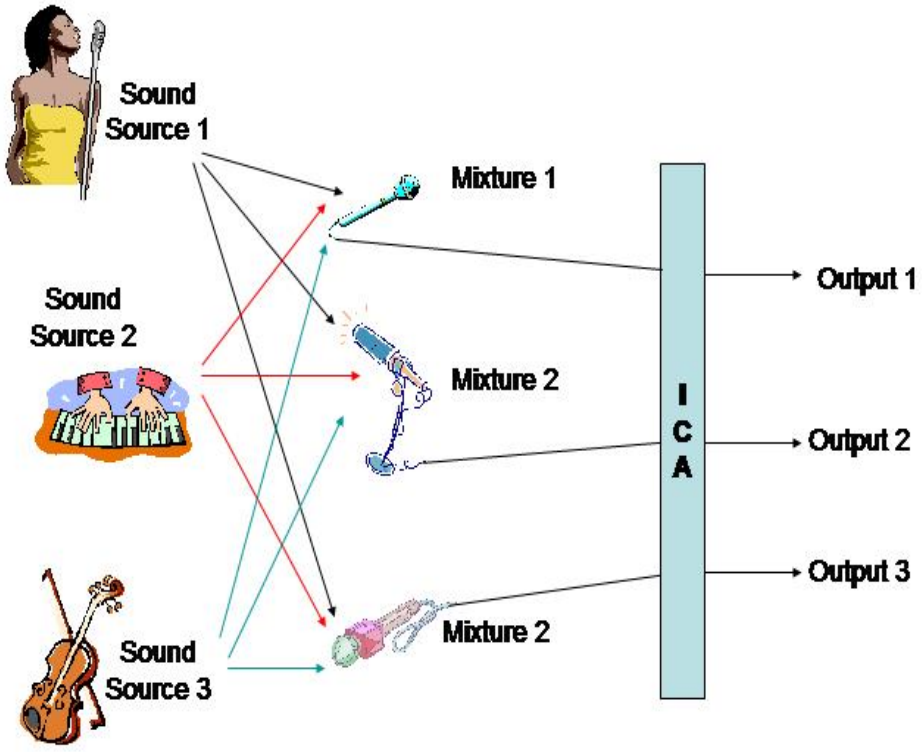

Figure 2.8: Cocktail party problem using ICA[6]

It can be determined from the source signals that the amplitude of one at any given point in time is completely unrelated to the others at that time. This indicates that they are produced by two different, unrelated processes. Knowing that the sources are unrelated, leads to the key strategy for ICA. Independent component analysis can recover the original

source signals by extracting the completely unrelated temporal signals of the two mixtures, and it is true that these extracted signals are the original unrelated voices[1].

2.4.2 Theoretical Background

As stated above, independent component analysis (ICA) is a higher-order statistical transform used in many of the same applications as principal component analysis. The biggest difference between the two being that independent component analysis is an estimation of components with a much stronger property then the components derived from PCA. Principle component analysis will extract a set of signals that are *uncorrelated*. If the set of mixtures in the cocktail party problem were microphone outputs then the extracted signals from PCA would be a new set of voice mixtures, where ICA would extract the independent signals from the mixtures. The independent signals would be a set of single voices[1]. These representations seem to capture the essential structure of the data in many applications, including feature extraction and signal separation[14]. The theory of ICA is based off of two primary equations seen in equation 2.8 and equation 2.9.

$$x = As \tag{2.7}$$

$$s = Wx \tag{2.8}$$

$$x = \sum_{i=1}^{n} a_i s_i \tag{2.9}$$

In the above equations x is the set of mixture signals, s is the independent source signals, and A is the mixing matrix used to create x out of a linear combination of the source signals as demonstrated by equation 2.9 where a_i is a column of A. The variable W is some unmixing matrix used to derive the source signals from the mixtures. This is the purpose of an independent component algorithm, to properly estimate an unmixing matrix that maximizes the nongaussianity, or statistical independence, of the source signals.

Higher-order processes like ICA use information on the distribution of x that is not contained in the covariance matrix. Because all the information of Gaussian variables

is contained in the covariance matrix, the distribution of x must not be assumed to be Gaussian[15]. For more general families of density functions, the representation has more degrees of freedom which the covariance matrix cannot account for. Much more sophisticated higher-level algorithms must be employed for non-Gaussian random variables. The transform defined by second-order methods like principal component analysis is not useful for many purposes where dimension reduction in that sense is not needed. This is because PCA neglects aspects of non-Gaussian data such as clustering and statistical independence of the components[15]. Principle component analysis derives uncorrelated components, which are not necessarily statistically independent. Statistically independent components are by definition, uncorrelated as well.

One fact about blind source separation methods like ICA is that there must be at least as many different mixtures of a set of source signals as there are source signals for the algorithm to yield the best results[1]. A very basic example of this is shown in figure 2.11, with the sawtooth and sine wave source signals. The first set of plots shows the source signals, the second shows two arbitrary linear mixtures of the source signals, and the third shows the extracted independent source signals after applying an ICA algorithm to the mixtures. If there are more source signals then signal mixtures, blind source separation algorithms have a great deal of difficulty extracting the source signals accurately, because the mixing matrix will not be the same dimensions as the estimated unmixing matrix. Typically, the number of signal mixtures is greater then the number of sources in practice. If this is the case, then the number of source signals extracted can be reduced by either specifying the number to extract or preprocessing the signal mixtures with a dimension reducing method like principal component analysis[1].

As can be seen by the plots of the extracted source signals, there are some ambiguities involved in performing independent component analysis: the order of the sources cannot be determined, the original amplitude of the sources can not be extracted, and the original phase of the signals cannot be determined either. None of these ambiguities, however, detract from the fact that the source signals have been separated out. The ambiguity of

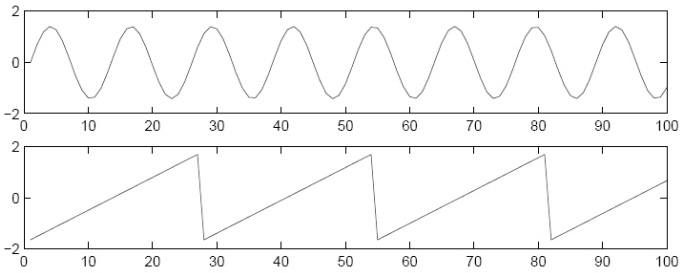

Figure 2.9: Example source signals[22]

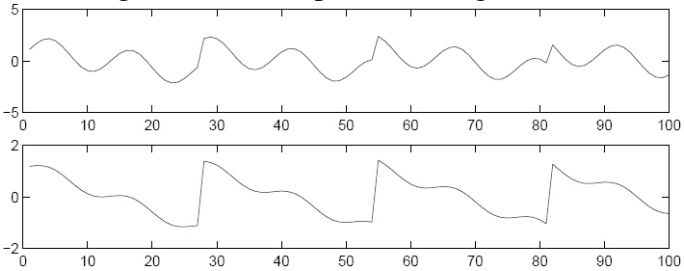

Figure 2.10: Example mixed signals[22]

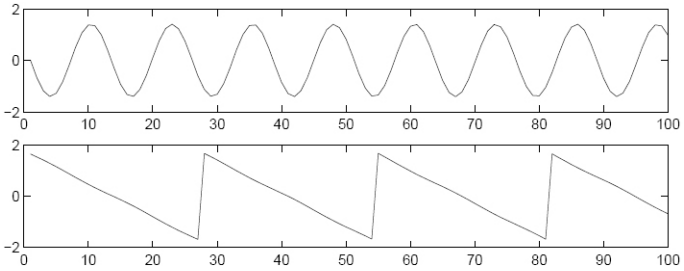

Figure 2.11: Sources extracted using ICA[22]

the order of sources should be obvious, since the order in which the sources were added can't be determined as long as the mixing matrix A and source matrix s are unknown. The second ambiguity is that the amplitude, or energies, of the sources is scaled by some factor in the extracted sources. This is again because A and s are unknown and any scalar multiplier in one of the sources s_i could always be canceled by dividing the corresponding column a_i of A by the same scalar. The most intuitive way to fix the magnitude to some value, is to assume that each has a variance equal to one. This leads to the third ambiguity of phase, or sign. Each independent component could be multiplied by -1 or 1 without

affecting the model. In most applications, however, these ambiguities are negligible or not features of the sources taken into account at all, and thus insignificant[14].

2.4.3 Independence

It should be clear by now that independent component analysis is based on this concept of statistical independence. Bascially, if two variables $y1$ and $y2$ are independent then the value of one variable provides absolutely no information about the value of the other variable[1]. For example, the temperature outside in Rochester, NY provides no information about who wins the Iditarod. These two events are independent of each other. Another example is tossing a coin. The result of each toss of the coin is independent of any other tosses, so the probability of getting *heads* on any number of tosses can be computed without taking *tails* into account. This discussion of independence can be extended to the more interesting example of speech. The probability that the amplitude of a voice signal s lies within an extremely small range around the value s^t is given by the value of the probability density function (pdf) $p_s\left(s^t\right)$ of that signal s^t[1]. Because speech signals spend most of their time with a near-zero amplitude, the probability density function (pdf) of a speech signal has a peak at $s = 0$, as shown in figure 2.12(left). The typical Gaussian probability density function associated with Gaussian data is depicted in figure 2.12(right).

Now, we assume that two speech signals s_1 and s_2 from two different people are independent. This implies that the joint probability is given by equation 2.10[1], where s^t is the pair of values in the speech signals at time t represented by $s^t = (s_1^t, s_2^t)$. Therefore the joint probability for all values of s is the joint pdf p_s, and can be visualized for two variables as shown in the left hand side of figure 2.12.

$$p_s\left(s^t\right) = p_{s1}\left(s_1^t\right) \times p_{s2}\left(s_2^t\right) \tag{2.10}$$

The pdfs p_{s1} and p_{s2} are known as the marginal pdfs of the joint pdf p_s. This probability density function can be obtained as the product of its two marginal pdfs only because the

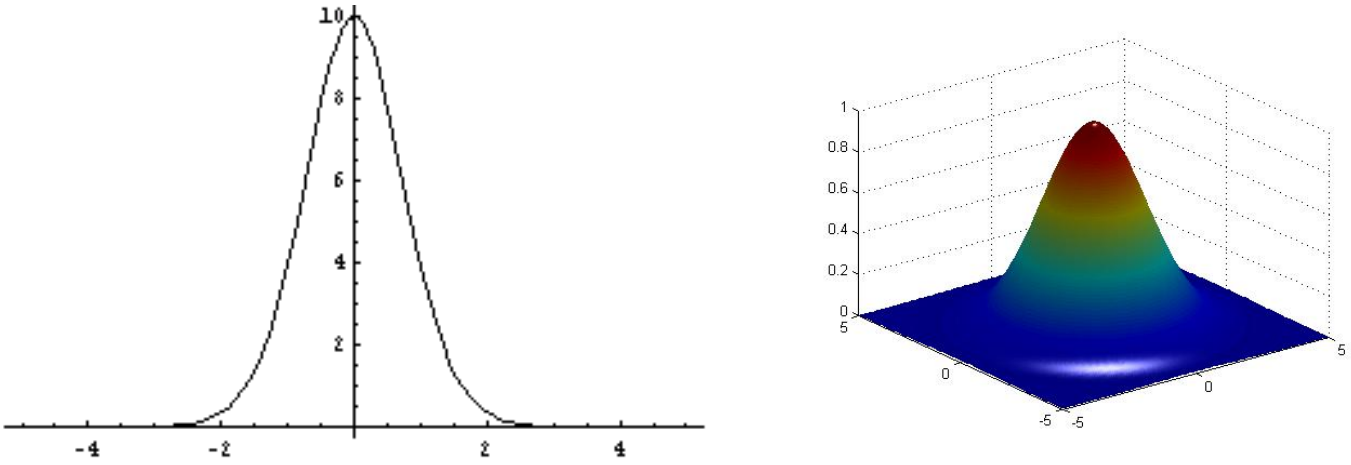

Figure 2.12: Speech joint PDF (left), Gaussian joint PDF (right)

variables s_1 and s_2 are independent. Just as the probability of flipping a coin and getting *heads* multiple times in succession can be obtained as the product of just the probability of getting *heads*. Due to the fact that it is assumed that all values of each speech signal are independent, the ordering of signal values is ignored[1].

The theory of statistical independence as applied to independent component analysis suggests that a method of extracting the source signals from a set of mixtures is to find the signals within the mixtures that are mutually independent, and therefore maximally non-Gaussian. A measure of how independent signals are is needed to do this, permitting an ICA algorithm to iteratively change the unmixing matrix in a way that increases the degree of independence. Two such measures of independence are negentropy and kurtosis.

2.4.4 Measuring Independence

The principles behind an independent component analysis algorithm is the estimation of sources and the maximization of statistical independence within the estimated sources. The fundamental goal of every ICA algorithm remains the same; maximize the nongaussianity

of the extracted sources. Referring back to the fundamental equations of independent component analysis (see equations 2.8 2.9 2.9), to estimate just one of the independent components, consider a linear combination of the x_i mixture values from equation 2.9; where $y = w^T x = \sum_i w_i x_i$, and w is a vector to be determined. If w was one of the rows of the inverse of A, this linear combination would equal one of the independent components[14]. Even though w can't be determined exactly, an estimation that gives a good approximation can be found[15].

To see how this leads to the basic principal of ICA estimation, a change of variables is made, defining $z = A^T w$. This gives $y = w^T x = w^T A s = z^T s$. y is therefore a linear combination of s_i, with weights (mixing matrix) given by z_i. Since the sum of even two independent random variables is more Gaussian than the original variables, $z^T s$ is more Gaussian than any of the s_i and becomes the least Gaussian when it equals one of the s_i[14]. Taking a vector w that maximizes the nongaussianity of $w^T x$ means that $w^T x = z^T s$ equals one of the independent components[14]. w has $2n$ local maxima, two for each independent component, corresponding to s_i and $-s_i$[14]. To find several independent components, we need to find all of these local maxima, which is not difficult because the different independent components are uncorrelated, so we can always constrain the search to the space that gives estimates uncorrelated with the previous ones[14].

These estimations require a measure of Gaussianity, or mutual dependence, with some of the most common being Kurtosis or negentropy. Negentropy will be addressed first, and is quite obviously based on the statistical concept of entropy. Entropy is a measure of the uniformity of the distribution of a set of values, so that complete uniformity corresponds to maximum entropy. If there exists a set of discrete signal values then the entropy of the set depends on how uniform the values are, and the entropy of this set of variables is known as the joint entropy[1]. An example of entropy for one variable and three variables can be seen in figure 2.13. In the plot on the left, entropy is graphed against the probability of a coin toss. When the probability of a coin toss is $p = 0.5$, the ability to predict it is minimal and so entropy is maximized. For three variables with a fixed range between zero and one.

Their entropy can be visualized as the degree of uniformity within the three dimensional plot shown.

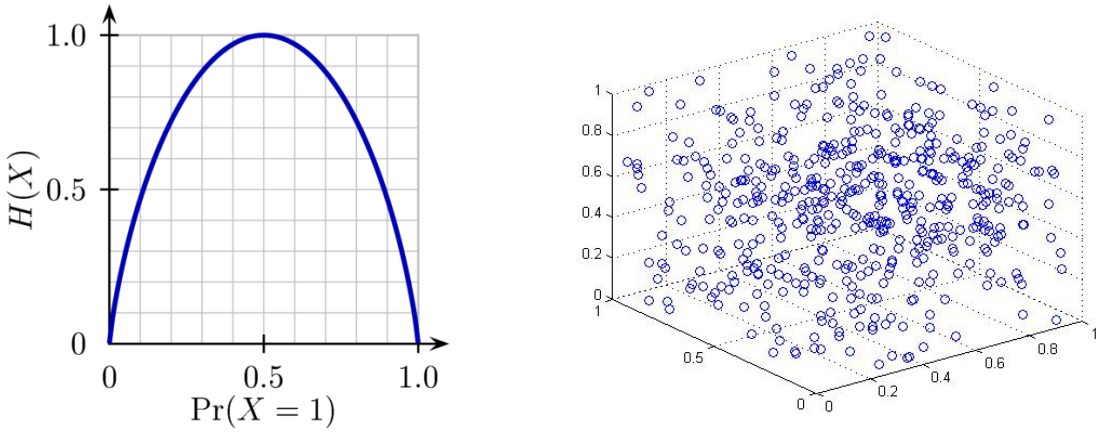

Figure 2.13: Graph of entropy relative to probability of a coin toss[1](left), plot of 3 variables with maximum entropy (right)

Entropy is the basic concept of information theory, the more "random" or unpredictable and unstructured the variable is, the larger its entropy. One way to calculate statistically independent components is to find the unmixing matrix that maximizes the entropy across all of the extracted signals. The differential entropy H of a random vector y with density $f(y)$ is defined in equation 2.11[14].

$$H(y) = -\int f(y) \log f(y)\, dy \tag{2.11}$$

A result of information theory is that a Gaussian variable has the largest entropy among all random variables of equal variance[14]. This means that entropy could be used as a measure of nongaussianity. Entropy is small for distributions that are clearly concentrated on certain values. To obtain a measure of nongaussianity that is zero for a Gaussian variable and always nonnegative, a slightly modified definition of differential entropy can be used, called negentropy given by equation 2.12[14],[15]

$$J(y) = H(y_{gauss}) - H(y) \tag{2.12}$$

where y_{gauss} is a Gaussian random variable of the same covariance matrix as y. Using these properties, negentropy is always non-negative, and it is zero if and only if y has a Gaussian distribution. The advantage of using negentropy, also called differential entropy, as a measure of nongaussianity is that it is well justified by statistical theory. The problem in using negentropy is, however, that it is computationally very difficult[14]. Approximations for calculating negentropy in a much more computationally efficient manner have been proposed as seen in equation 2.13. Varying the formula's used for G can provide further approximation with a minimal loss of information.

$$J(y) \approx \sum_{i=1}^{p} k_i \left[E\{G_i(y)\} - E\{G_i(v)\}\right]^2 \tag{2.13}$$

This approximation of negentropy gives a very good compromise between the two classical measures of nongaussianity, kurtosis and negentropy. They are conceptually simple, fast to compute, and still have desired statistical properties. This approximation is a good place to introduce the other measure of nongaussianity, used in a method called projection pursuit, kurtosis. The kurtosis of a signal is defined by equation 2.14, but in practice is computed by equation 2.15[1].

$$K(y) = E\left\{y^4\right\} - 3\left(E\left\{y^2\right\}\right)^2 \tag{2.14}$$

$$K(y) = \frac{\frac{1}{N}\sum_{t=1}^{N}\left(\overline{y} - y^t\right)^4}{\left(\frac{1}{N}\sum_{t=1}^{N}\left(\overline{y} - y^t\right)^2\right)^2} - 3 \tag{2.15}$$

According to equation 2.14, since y is assumed to be of unit variance, the equation simplifies to $E\{y^4\} - 3$. Kurtosis is then a normalized version of the fourth moment $E\{y^4\}$. For a Gaussian y, the fourth moment equals $3\left(E\{y^2\}\right)^2$. Thus, the kurtosis is zero

for a Gaussian random variable, whereas most nongaussian random variables will have a nonzero kurtosis (either positive or negative). Random variables that have a negative kurtosis are called sub-Gaussian, and those with positive kurtosis are called super-Gaussian. The magnitude of a signal's kurtosis has been widely used as a measure of nongaussianity in ICA. The main reason is its simplicity; computationally, kurtosis can be estimated simply by using the fourth moment of the sample data, seen in equation 2.14[14]. The computationally practical equation for kurtosis (equation 2.15) has only one important term, the numerator. The other terms are to take into account signal variance[1].

2.4.5 ICA and Projection Pursuit

Projection pursuit is a technique developed in statistics for finding interesting projections of multidimensional data. These projections can then be used for optimal visualization of the data, and for such purposes as density estimation and regression[14]. It has been argued by [10] and others in the field of projection pursuit, that the Gaussian distribution is the least interesting one, and that the most interesting projections are those that exhibit the least Gaussian distribution. This is almost exactly what is done during the independent component estimation of ICA, which can be considered a variant of projection pursuit[14]. The difference is that projection pursuit extracts one projected signal at a time that is as nongussian as possible, whereas independent component analysis which extracts M signals from M signal mixtures simultaneously[1].

Specifically, the projection pursuit allows us to tackle the situation where there are less independent components s_i than original variables x_i. However, it should be noted that in the formulation of projection pursuit, no data model or assumption about independent components is made[14]. In ICA models, optimizing the nongaussianity measures produces independent components; if the model does not hold, then the projection pursuit directions are produced[4].

2.4.6 Preprocessing

Before employing any independent component algorithm, it is most often useful to do some preprocessing on the data set. This section will briefly describe a few preprocessing techniques commonly used in conjunction with ICA, such as centering and whitening. The most basic and necessary preprocessing is to center x by subtracting its mean to make x a zero-mean variable. This preprocessing is made solely to simplify the ICA algorithms, but they will calculate the actual mean otherwise. After estimating the mixing matrix A with centered data, we can complete the estimation by adding the mean vector of s back to the centered estimates of s. The mean vector of s is given by $A - 1m$, where m is the mean that was subtracted in the preprocessing[14].

The other useful preprocessing tool is data whitening, which occurs before performing the independent component analysis algorithm, but after centering. The observed vector x is linearly transformed so that it is *white*. A vector is white when its components are uncorrelate and their variances equal unity[14]. The result of doing this is a data vector whose covariance matrix is equal to the identity matrix, given by equation 2.16.

$$E\left\{\tilde{x}\tilde{x}^T\right\} = I \tag{2.16}$$

One popular method for whitening is to use the eigen-value decomposition (EVD) of the covariance matrix. Whitening transforms the mixing matrix into a new one, $\tilde{A}$ giving equation 2.17 (see [14] for proof).

$$\tilde{x} = ED^{-\frac{1}{2}}E^T As = \tilde{A}s \tag{2.17}$$

Whitening reduces the number of parameters that need to be estimated, from n^2 parameters of the original matrix A to $n\left(n-1\right)/2$ parameters of the orthogonal mixing matrix $\tilde{A}$[14]. Whiteneing essentially reduces the complexity of the problem that has to be addressed by the ICA algorithm, saving on computational resources[4].

2.4.7 FastICA Algorithm

The FastICA algorithm was developed at the Laboratory of Information and Computer Science in the Helsinki University of Technology by Hugo Gvert, Jarmo Hurri, Jaakko Srel, and Aapo Hyvrinen. The FastICA algorithm is a highly computationally efficient method for performing the estimation of ICA. It uses a fixed-point iteration process that has been determined, in independent experiments, to be 10-100 times faster than conventional gradient descent methods for ICA. Another advantage of the FastICA algorithm is that it can be used to perform projection pursuit as well, thus providing a general-purpose data analysis method that can be used both in an exploratory fashion and for estimation of independent components (or sources)[3].

The basic process for the algorithm is best first described as a one-unit version, where there is only one computational unit with a weight vector w that is able to update by a learning rule. The FastICA learning rule finds a unit vector w such that $w^T x$ maximizes non-gaussianity, which in this case is calculated by the approximation of negentropy $J\left(w^T x\right)$. The steps of the FastICA algorithm for extracting a single independent component are outlined below.

- Take a random initial vector $w(0)$ of norm 1, and let $k = 1$
- Let $w\left(k\right) = E\left\{x\left(w\left(k-1\right)^T x\right)^3\right\} - 3w\left(k-1\right)$. The expectation can be estimated using a large sample of x vectors (say, 1,000 points).
- Divide $w\left(k\right)$ by its norm.
- If $\left|w(k)^T w\left(k-1\right)\right|$ is not close enough to 1, let $k = k + 1$ and go back to step 2. Otherwise output the vector $w(k)$.

After starting with a random guess vector for w, the second step is the equation finding maximum independence, with the third checking the convergence to a local maxima. Derivations of these steps can be found in [13]. The final $w(k)$ vector produced by the FastICA algorithm is one of the columns of the orthogonal unmixing matrix W. In the case

of blind source separation, this means that $w(k)$ extracts one of the nongaussian source signals from the set of mixtures x. This set of steps only estimates one of the independent components, so it must be run n times to determine all of the requested independent components. To guard against extracting the same independent component more then once, an orthogonalizing projection is inserted at the beginning of step three, changing it to the item below.

- Let $w\left(k\right) = w\left(k\right) - \overline{WW}^{T} w\left(k\right)$

Becuase the unmixing matrix W is orthogonal, independent components can be estimated one by one by projecting the current solution $w(k)$ on the space orthogonal to the columns of the unmixing matrix W. The matrix $\overline{W}$ is defined as the matrix whose columns are previously found columns of W. This decorrelation of the outputs after each iteration solves the problem of any two independent components converging to the same local maxima[13].

The convergence of this algorithm is *cubic* (see [13] for proof), which is unusual for an independent component analysis algorithm. Many algorithms use the power method, and converge linearly. The FastICA algorithm is also hierarchical, allowing it to find independent components one at a time instead of estimating the entire unmixing matrix at once. Therefore, it is possible to to estimate only certain independent components with FastICA if there's enough prior information known about the weight matrices[13]. The FastICA algorithm was developed to make the learning of kurtosis faster, and thus provide a much more computationally efficient way of estimating independent components. It's performance and theoretical assumptions have been proven in independent studies and [13], and given cause for it to be used in the development of a process that will quickly evaluate and separate the high-level components of acoustic information. The originally developed FastICA algorithm Matlab implementation can be found at *http://www.cis.hut.fi/projects/ica/fastica/*, with implementations in other languages existing elsewhere[3].

2.4.8 Temporal vs. Spatial ICA

For all of the independent component analysis examples given so far, a set of M temporal signal mixtures is measured over N time steps, and M temporal source signals are recovered so that each source signal is independent over time of the others[1]. When considering time-domain signals, each row of the data matrix x is a temporal signal over time. The data can also be viewed as the transpose where each column of the original matrix x is a spatial signal at one point in time. When the rows of the matrix x are treated as mixtures over time, ICA produces temporal independent components, but treating the columns as mixtures produces spatial independent components[1]. Independent component analysis can therefore be used to maximize independence over time or space. An example of each type of independent component analysis can be given in speech signal separation (cocktail party problem) for temporal ICA and functional magnetic resonance imaging (fMRI) for spatial ICA. For the rest of this section comparing the two forms of ICA, sICA will be used to refer to spatial independent component analysis and tICA will be used to refer to temporal independent component analysis[1].

The problems solved by tICA and sICA are, for the most part, the same and the algorithms used are certainly the same. The only difference is in the orientation of the mixture matrix passed to the ICA algorithm, and whether the extracted components are considered independent over time or over space.

tICA

Temporal independent component analysis is used to extract each temporal source signal from the mixtures at each point in time, but only one point in space. This is much better explained through the example of blind source separation to solve the cocktail party problem. Each n-length mixture signal comprises a row of the overall mixture matrix and can then be viewed as a variable, with n observations at successive points in time[1]. Assuming m different mixtures, then ICA will try to extract signals that are mutually independent at every

observed point in time producing an n by n unmixing matrix. The unmixing matrix coupled with the mixture signals will yield an m by n independent component matrix, where each of the m rows are mutually independent temporal signals. Each of the independent temporal components defines how a specific source contributes to the temporal sequence of mixtures at each point in time.

sICA

Spatial independent component analysis can be interpreted as each signal being a mixture of underlying source signals, and a source vector as a spatial sequence measured at a single point in time. This is in direct contrast with tICA, which considers temporal mixtures or underlying temporal source signals measured over time and at one point in space[1]. Using the same example as tICA, but with the mixture matrix transposed (i.e. n rows and m columns) to place each observation of the temporal signals in a row, sICA will determine the set of independent spatial source signals. Thus, sICA would estimate an m by m unmixing matrix so that the extracted spatial signals are mutually independent. This will output a set of m-length independent components that are independent over space of spatial patterns at all other points in time[1].

2.4.9 Use in Cognitive Modeling

As stated earlier, independent component analysis has been used in many applications in the field of computer vision and signal processing because of its success in identifying underlying components and patterns in very complex, multi-dimensional data. ICA has found uses in many different areas of cognitive processing, such as neuroscience, statistical analysis, physics, audio signal processing, and image and video processing. This section will briefly outline some of the applications for using independent component analysis to develop cognitive processes, and introduce some more of the ideas that make up the basis for this research.

In the field of neuroscience, some of the most important tools available is functional

magnetic resonance images (fMRI) and Magnetoencephalography (MEG). These are both noninvasive techniques by which the activity of the cortical neurons can be measured with a very good temporal resolution and moderate spatial resolution. When using an MEG record, the user will most likely be faced with the problem of extracting the principal features of the neuromagnetic signals adjusting for the presence of artifacts. In [12], the authors introduced a new method to separate brain activity from artifacts using independent component analysis, specifically sICA[14]. The approach is based on the assumption that the brain activity and the artifacts are anatomically and physiologically separate processes, and this separation is reflected in the statistical independence between the magnetic signals generated by those processes. ICA has been proven effective in extracting spatially independent features of interest from the MEG and fMRI records[14] [12]. This is an application of sICA for a process that is not intuitive to human beings, like many of the applications with images and sound are.

In the statistical analysis world, more often then not associated with commerce and finance, it is a tempting alternative to try independent component analysis on financial data. There are many situations in which parallel time series are available, such as currency exchange rates or daily returns of stocks, that may have some common underlying factors. ICA might reveal some driving factors of the data that otherwise remain hidden. The assumption of having some underlying independent components in this specific application is not unrealistic[14]. For example, factors like seasonal variations due to holidays and annual variations, and factors having a sudden effect on the purchasing power of the customers can be expected to have an effect on all markets. Such factors can be assumed to be roughly independent of each other. Depending on the policy and skills of the individual retailer, such as advertising efforts, the effect of the factors in the overall market on particular businesses are slightly different. By ICA, it is possible to isolate both the underlying factors and the effect weights, therefore also making it possible to group the stores on the basis of their managerial policies using only the cash flow time series data[14].

In [17], ICA is used twice within an image processing algorithm to attempt to learn

the ”gist” of a set of natural images and group them based on their higher level features. Independent component analysis has two quite common roles in image processing, extracting low-level features, and matching images to the patterns in them. In this experiment, a set of natural images were passed through independent component analysis to extract the independent features throughout the entire data set of images. In the next step, the images were individually passed through ICA to extract the independent feature specific to that image[17]. This allowed the images to be grouped together based on which of the features generated in the first step best matched features from the second. This leads to content-based grouping of images whose high level features are extremely similar. Results of this implementation can be seen in the image groupings below.

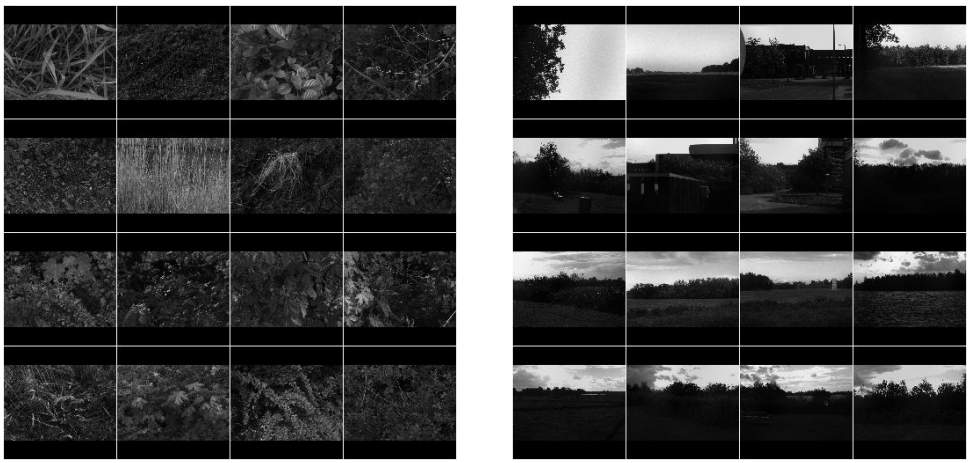

Figure 2.14: Grouping of underbrush images(left) and horizons(right) [17]

A good survey of independent component analysis applications can be found in [16] where principal and independent component analysis is used to demonstrate possible useful applications in text analysis, analyzing social networks, and musical genre analysis. This work developed processes for various forms of text, social network, and musical genre analysis using principal component analysis. The eigenvectors generated, and their relationships to each other were shown to exhibit characteristics of statistical independence. This leads to the proposition that processes exist using independent component analysis

to exploit this independence for exploring automatic text content analysis, social network understanding, and musical genre classification. It is the processes posed in this work that have, in great part, lead to the research developed in this thesis. The idea of developing a process to analyze music files in order to automatically determine their respective genres is also explored in [24] [20] [18]. This work will attempt to develop a process, based on independent component analysis, to automatically segregate acoustic information, including not just music, but world languages as well.

Chapter 3

Cognitive Model

3.1 Previous Work

The basis for this thesis research has come from the current significance of the problems that exist in automatically labeling, categorizing, and managing digital media expressed in the introduction. Extensive research was done to explore the advancements and previous work that had been done in this area. I looked into the subject matter, background theory, processes, and signal processing techniques being used to address these issues and develop a specific area of research to focus the novel work presented here. After a great deal of searching, it was determined that a quick, computationally efficient process using some common preprocessing method and independent component analysis to separate general acoustic information did not yet exist. This section will discuss the previous work done in this area, and the extensions to that work used to comprise the process developed in this thesis.

In previous works in the area of acoustic feature extraction and modeling, it has been shown that Mel-frequency cepstral coefficients are a good audio preprocessing step to extract general low-level features from music and speech signals [16] [18]. The windowed MFCC's create a smoothed signal that highlights strong temporal frequency responses without losing information from the original signal. Also, reasonable decision time horizon's have been determined in similar methods used for audio classification, which attempt to find the minimal time sample of audio signals that is required to make a good estimation

of the entire signal's classification[20]. This is an important feature in developing a signal processing algorithm. In practice, an algorithm developed for the automatic organization of digital media should execute in a timely and efficient manner. To analyze the entire audio signal, which can vary greatly in length, would take an excessive amount of time. It's also intuitive to the end goal of the process, because humans can determine the general classification of an audio signal after hearing only a short portion of it. To only use a small portion of the signal in classification is not only biologically accurate, but computationally efficient as well.

However, the Mel-frequency cepstral coefficients of a sample audio signal is not a high enough abstraction of the original signal to perform classification. MFCC's are effectively coefficients of a signal's relative response to certain frequency bands. Therefore, using another form of signal processing to find the higher-level temporal themes in the signal would give a good representation of the temporal patterns and components of the input signals. This representation of patterns across high dimensional data will yield a set of themes that should be similar across signals within the same class and distinct across signals of different classes.

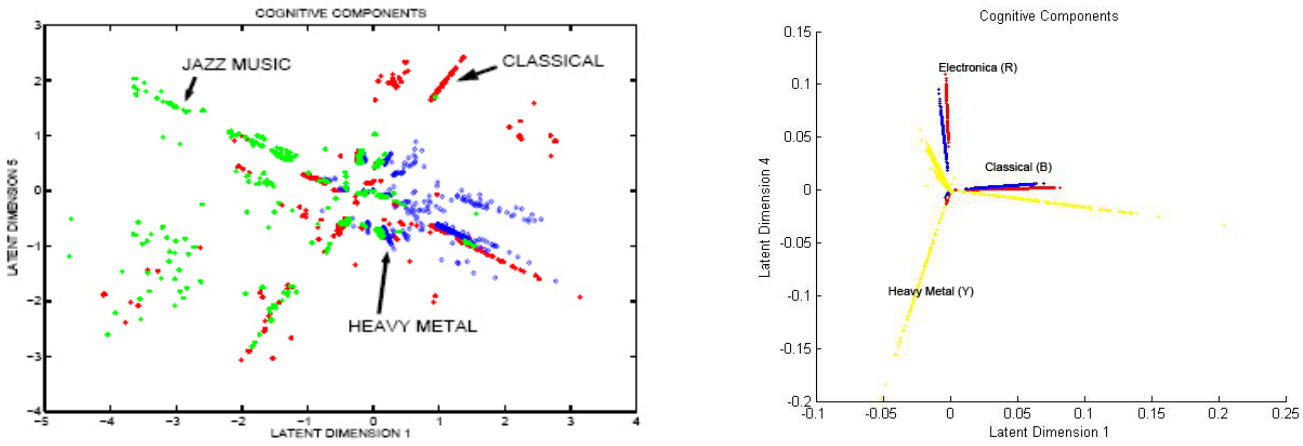

Figure 3.1: Music preprocessing results from [16] (left), and recreated results (right)

The authors of previous works used principal component analysis to demonstrate that

different genres of music would produce a relatively unique set of characteristic themes. As can be seen in figure 3.1, the author's original work using principal component analysis showed that on some higher dimensions, the music occupied different spaces and even displayed some statistical independence. To gain some understanding of these results and the feasibility of the work, I have recreated some simple results using PCA and placed them next to the plot of the original work. This work, as well as the work of others in the area of acoustic classification has shown that the use of statistical transforms like PCA and ICA, for signal processing, is a reasonable means for extracting unique and significant patterns from acoustic data [16] [24] [20]. It was proposed in [16] that because the features of different genres of music showed signs of statistical independence, that the use of a combination of Mel-frequency cepstral coefficients and independent component analysis in classifying music by genre and sub-genre would be a good extension to the concepts they had demonstrated. This extension into developing a process for separating classes of music, and further extension to separate different kinds of speech is the primary inspiration for the research work done in this thesis.

3.2 Process Development

3.2.1 Preprocessing

The steps of this process have been initially based on the previous works studied and also the proposed additional processing using independent component analysis. The first part in the process is to determine what data preprocessing will take place to aid the ICA algorithm in extracting the most significant temporal themes from the acoustic data. It's reasonable that since most acoustic information, whether it be speaking voices or music, maintain certain constant characteristics regardless of amplitude or time, that any random sample of significant length in the audio file will contain the same properties as the entire file [20]. The low level features of this sample can then be extracted using Mel-frequency cepstral coefficients, as discussed earlier. This information will give us a set of features for

each short window of time within the sample to include in further processing. Figure 3.2 illustrates the acoustic preprocessing for each audio sample introduced into the system.

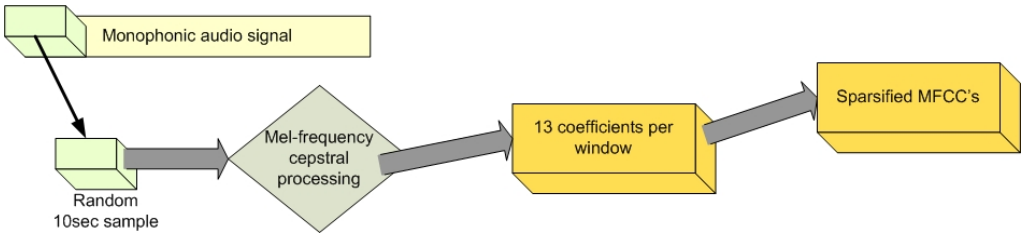

Figure 3.2: Acoustic preprocessing of one piece acoustic data

Before organizing the data from each audio file and proceeding to ICA, further preprocessing of the Mel-frequency cepstral coefficients could take place to make the calculation and feature extraction of ICA easier. The preprocessing steps discussed in [14] are not necessary at this stage since the most recent version of the FastICA algorithm (v2.5 at the time of this writing) includes steps for both data centering and whitening by way of eigen decomposition. However, the MFCC's will be sparsified to make the signal independent of pitch, and only retain the most significant coefficients to be passed to ICA[16]. The MFCC's were sparsified by retaining onl those coefficients with a magnitude in the top 5% for each frequency band.

3.2.2 ICA for Feature Extraction

The next part of the process will be to try to isolate its high-level temporal features using independent component analysis. Because the goal is to demonstrate uniqueness amongst classes of acoustic information, the samples will be organized into pre-determined groupings. In the case of these experiments, the groups correspond to genres of music or world languages. The respective Mel-frequency cepstral coefficients of each sample within a class will also be grouped together. This is done because the MFCC's extracted from a single piece of audio data are relative to different frequency responses of the same physical

process. To extract the temporal themes of a particular class of acoustic data, a particular MFCC for all the audio samples in a particular class are grouped together before being processed with independent component analysis. These sets of coefficients are then passed through ICA to extract the most significant themes or patterns in the coefficients for that group. Figure 3.3 depicts the processing for one MFCC for one group of audio data.

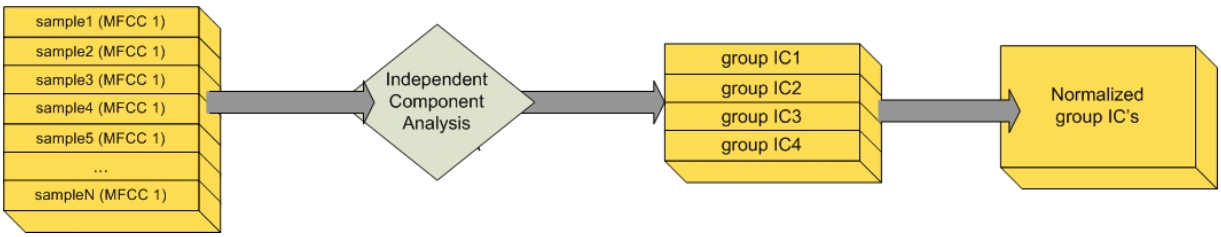

Figure 3.3: Training process for a single MFCC for a single group

Some very basic normalization of the independent components to correct for the ambiguities in amplitude is applied after the ICA algorithm. The result of the process depicted in figure 3.3 is a matrix of the N independent components, representing underlying temporal features, of one of the Mel-frequency cepstral coefficients of one class of audio data. This process is repeated for each MFCC for all of the classes of acoustic data. For an implementation that uses c MFCC's and m classes of acoustic information, this process will yield mc matrices of N components each. The significance of organizing data this way, is that each class of audio data will have cN points of comparison. So that any new information processed through the system in a similar manner should be able to be compared against all the points of comparison in each of the classes of acoustic data. The class that contains the greatest number of similar points of comparison will be the class into which the new information could be classified. Figure 3.4 shows the collection of matrices that make up the points of comparison for each class of audio data. The feasibility of automatically classifying new data comes in demonstrating the uniqueness of these points of comparison across the different classes.

The final part of the process is to use these c points of comparison for each class of

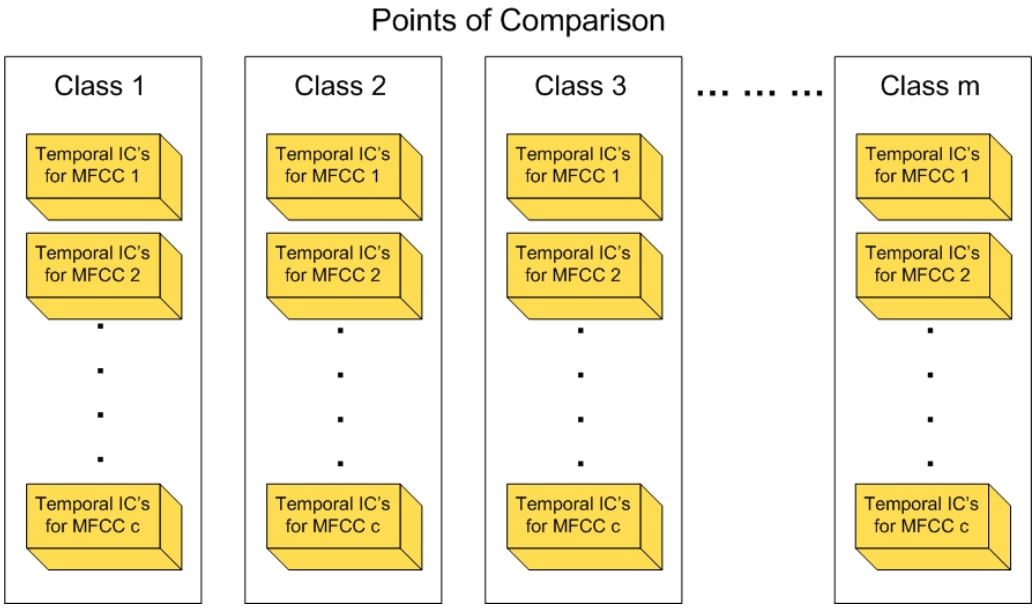

Figure 3.4: Points of comparison for classes of acoustic data

acoustic information to demonstrate the uniqueness of the classes. The c sets of independent temporal components of each set of Mel-frequency cepstral coefficients for one class will demonstrate some unique separation of underlying sources from the same sets in other classes. The next section will explain the specifics of how this process was implemented to expose this uniqueness.

Chapter 4

Process Implementation

This process of acoustic information classification was implemented using the Matlab language to perform all of the matrix operations and other data manipulation used in this research. Additional statistical processing toolboxes were used, which provided functions for producing the relevant information associated with audio processing, Mel-frequency cepstral coefficients [8], and independent component analysis[3]. All of the prototyping and implementation was completed in Matlab's programming and scripting language, and can be found (fully commented!) in Appendix A.

4.1 Data Set

The focus of this thesis research is to use the process described in the last section to demonstrate the separation of musical genres and/or sub-genres based on their independent temporal components extracted using Mel-frequency cepstral coefficients and independent component analysis. Then extending that separation to a set of world language samples to measure the generality and robustness of the process. In order to perform an adequate number of experiments and fully exercise the independent component algorithm, a large database of acoustic files were compiled for the research. Because the inspiration for this work is the automatic processing of digital media, the database consisted of mp3 files which is the dominant format. Varying parameters of the mp3 files in the database, such as sample rate, length, meta data (i.e. ID3 tags, file names, etc.) are accounted for in the implementation of

the process and were not considered when compiling the data. This is a real-life variance in data quality and content that must be taken into account for this process to have practical applications.

4.1.1 Test Tones

The initial set of data to be tested with this system, in order to correlate the appearance of the results with the acoustic features they represent, was a collection of test tones. Test tones are widely used in sound calibration for all kinds of audio systems, and a number of samples and tests of these tones exist. The tones include bursts of constant frequency and constant amplitude with some intermittent silence, pulses of constant frequency and constant amplitude, but no intermittent silence, and ramps of varying frequency and constant amplitude. An example of some of these tones can be seen in figure 4.1. These tones helped to determine what the features and underlying temporal signals for the process correspond to in the source audio signal.

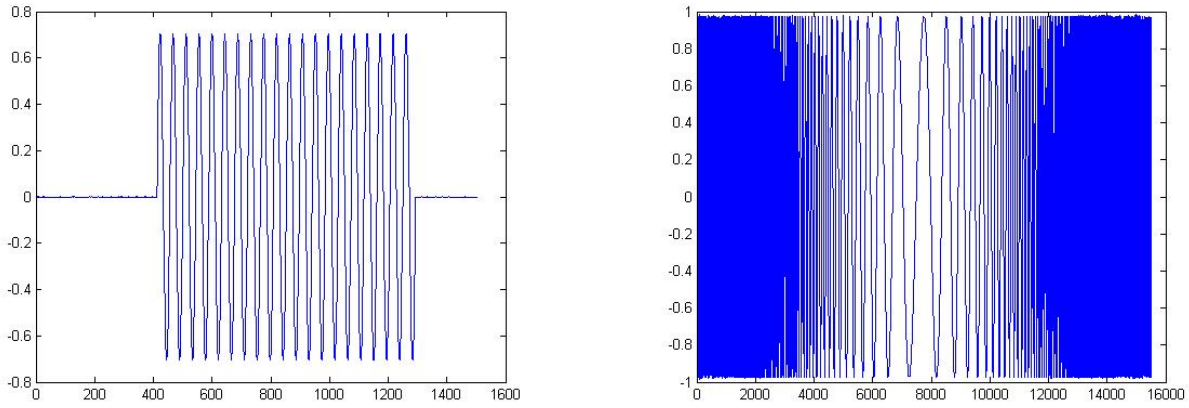

Figure 4.1: Example test tones, burst (left) and ramp (right)

4.1.2 Music Genres

In order to yield the best results, four genres, that intuitively sound as different as possible, were selected to comprise the data set. They were classical music, jazz, electronic (techno), and heavy metal, which includes the three genres used in the work done by [16] for some continuity between the previous work and this extended research. One hundred mp3 files, varying in length, but at least one minute long, were compiled for each particular genre of music based on compilation CD's that provide a good representation sample of each style of music. In total, four hundred songs of significant length and varying artists.

4.1.3 World Languages

For the database of spoken world languages, some more work was involved as I could not find a readily available source of the type of speech samples desired for this work. The spoken world language database was recorded in a sound studio from ten different volunteer, bi-lingual subjects of varying countries of origin, so each subject could speak both English and their native language. Because of the controlled nature of the recording environment, the spoken languages are completely devoid of arbitrary noise or other acoustic interference. Each subject was first recorded speaking the same arbitrary paragraph of text in English, yielding a speech sample approximately one minute in length. This information is important for the purposes of this work, because it will help distinguish if the process separates out acoustic information based on the world language being spoken or simply the characteristics of the speaker's voice.

After recording the speech in English, each subject then recorded another minute long speech segment speaking some arbitrary piece of text in their native language. Recordings were made for English (5 speakers), Hindi (5 speakers), German (1 speaker), Dutch (1 speaker), Bulgarian (1 speaker), Korean (1 speaker), and Polish (1 speaker). While this database is significantly smaller then that of the musical genres, it was sufficient for a proof of concept evaluation of the developed process once it had been successfully demonstrated

with the musical genre database.

4.2 Preprocessing

4.2.1 Random Sampling

For the purposes of demonstrating separation between classes of acoustic information, the entire music database does not need to be processed. The rest of this section will describe the process being implemented on forty music samples from the database for simplicity, ten from each of the four genres. The implementation of this process is completely scalable and *has* been run on the entire database to ensure this is true. As described earlier, the first step in the process is to extract a random, representative sample from each of the files being used in the experiment. The size of the sample being extracted was chosen to be ten seconds, as that was determined to be a good decision time horizon for music classification by [20]. The effect of changing the size of the sample is that too large a sample will become computationally expensive and take an excessive amount of time, as well as possibly causing the ICA algorithm to encounter over learning [1] and perform more calculations then necessary. If the size of the sample is too short of a time frame, then not enough acoustic information will be captured to reflect the higher level temporal characteristics of the entire file[1].

Each sample vector is read using the *mp3read* function from the audio processing toolbox[8] across the randomly selected interval of time at a sample rate of $\tilde{2}$2K samples per second (22,050 samples/sec). If the signal is a stereophonic audio file, it's converted to a monophonic audio file so that each 10 second sample of acoustic information is represented by a 220,500 length vector. This vector is the sample single that will be grouped with other samples based on its acoustic class and organized in a 2-D matrix. Each audio class matrix will have a length of 220,500 and 10 rows, one for each sample in the group. Each of these matrices are then concatenated in the third dimension and passed to the second step of the process, which is deriving MFCC's for each sample signal as seen in figure

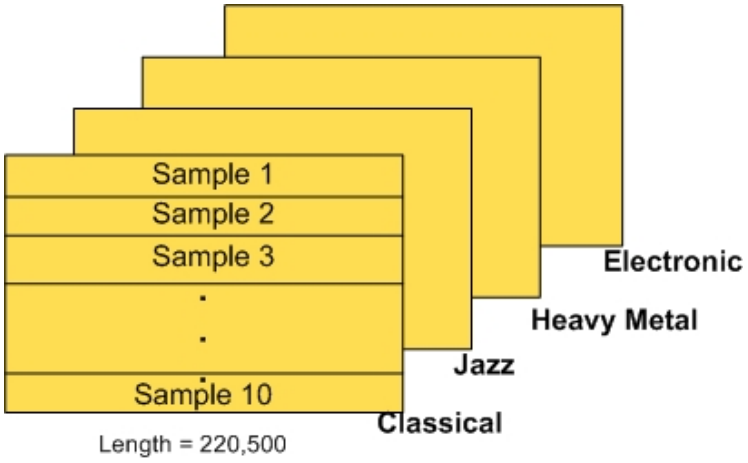

Figure 4.2: Grouping of audio samples

4.2.

4.2.2 Calculating MFCC's

The process for calculating each sample vector's Mel-frequency cepstral coefficients was outlined in section 2.2, but in practice can be done with the use of the auditory toolbox[8], which includes functions to produce MFCC's for a vector signal with a known sampling rate. Changing the parameters in the Mel-frequency cepstral coefficients algorithm has some consequences associated with it, both positive and negative. Adjusting the windowing time size and overlapping time size controls how much smoothing over time occurs and how much redundant information is retained respectively. A windowing size that is too large will generalize that time frame in the sample to one coefficient, losing some of the variance in that signal over that time frame. This causes a loss of detail in the low-level features that can propagate to a loss of detail in the temporal patterns extracted by ICA. If the overlap time was zero milliseconds, then each time window would be extracted end to end neglecting features that may span across them. Creating an overlap time allows these features to be captured by at least one window. The downside to overlap is that it causes some redundancy, because multiple windows will use the same part of the sample signal in

their calculations.

The other parameter to generating the Mel-frequency cepstral coefficients is the number of coefficients to calculate per time window. In the area of speech processing and recognition, 13 coefficients is the unofficial standard. This standard for arranging the analysis of frequency ranges of human speech is intended to mimic the response of the human ear. Therefore, it is reasonable to apply this same standard to applications of music and other acoustic information, as has been done in previous works [7] [9] [22].

The 13 Mel-frequency cepstral coefficients are derived at each point in time for a particular audio signal. The input signal is windowed into 30ms windows with 10ms of overlap [16] [20] Thus, the output is a 13 by 1000 matrix of cepstral coefficients for each input signal that exemplifies the low-level features of that signal. After the Mel-frequency cepstral coefficients have been calculated, they are sparsified within each signal to make them independent of pitch and leave only the coefficients that have the greatest relative response in the sample. Sparsifying is done by maintaining only the top 5% magnitude values for each point in time[16]. The matrix retains its size of 13 x 1000 because any coefficient value not in the top 5% magnitude fractile is simply changed to 0.

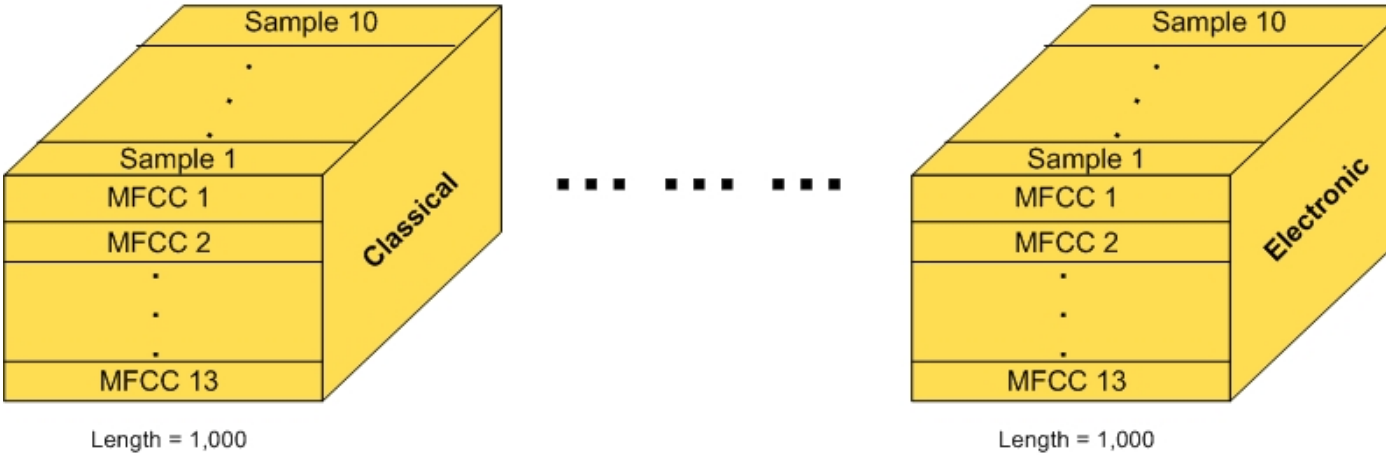

Figure 4.3: Grouping MFCC's for each signal and all groups into feature matrices

At this point in the process, there is now a two dimensional matrix for each sample

containing its sparsified Mel-frequency cepstral coefficients. The sparsified MFCC's for each sample within a group are concatenated on the third dimension, forming the low-level feature matrix with a size of 13 by 1000 by 10. Figure 4.3 depicts the orientation and organization of the low-level feature matrices for the groups. Once the feature matrix for each group is completed, the temporal themes of the sparse MFCC's are ready to be extracted using independent component analysis. The low-level feature vectors of each group were then concatenated on the fourth dimension and passed in one variable to the next part of the process for temporal feature extraction.

4.3 Independent Feature Extraction

The second part of this process is the independent feature extraction, which will take the low-level features of the groups of samples and use independent component analysis to extract out the underlying temporal features from them. For this part, the Matlab implementation of the FastICA algorithm[3] was used for feature extraction. The FastICA algorithm contains a number of optional parameters that can be used to adjust the algorithm, however the default values of these parameters are usually a good use of the algorithm's features and robustness. Based on the best results obtained from this process using the FastICA algorithm, *tanh* nonlinearity was used for the $G_i(y)$ in equation 2.13. The decorrelation approach used can be *symmetric*, which estimates all of the independent components simultaneously, or *deflation* which will estimate them one at a time like in projection pursuit. These were used interchangeably with a negligible change in the output, so for the purposes of this description symmetric decorrelation is used. The last two parameters changed from their default were *lastEig* and *numOfIC*, which set the last eigenvector to keep during eigenvalue decomposition and the number of independent components to return, respectively. The *lastEig* parameter was set to 6 to drop out the 4 least significant eigenvectors as they won't contribute greatly to accounting for variance in the data, and *numOfIC* to 5 to reduce the dimensionality of the 10 input signals.

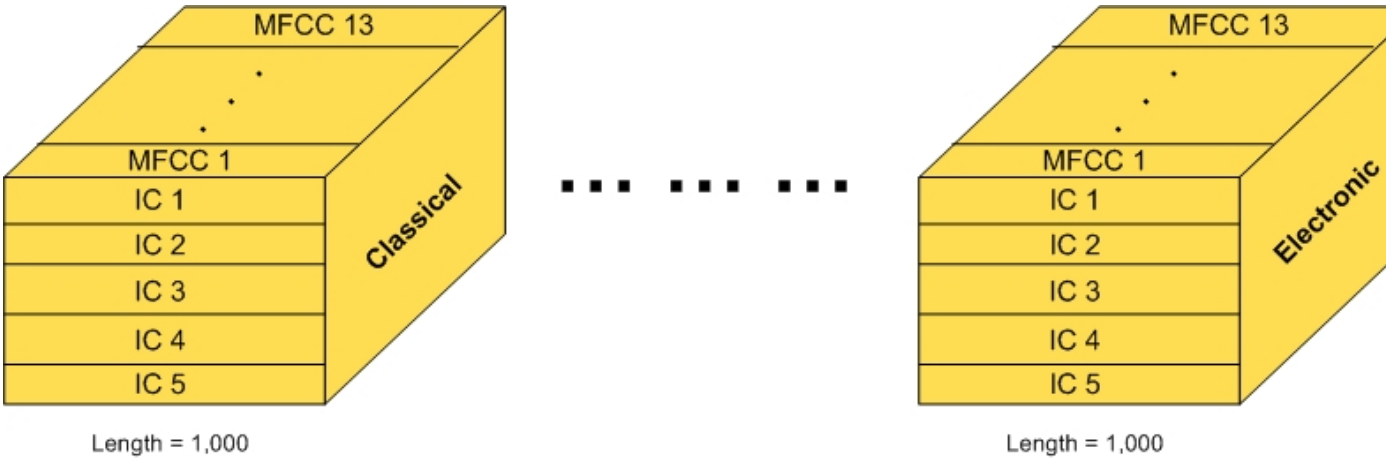

Figure 4.4: Grouping of independent components into points of comparison for the groups of acoustic data

The manner in which each Mel-frequency cepstral coefficient, across all audio samples, is passed to the FastICA algorithm was outlined in figure 3.3. The collection of first MFCC's for each sample are passed through the FastICA algorithm with the parameters given above, then the collection of second MFCC's, then the third, and so on for all 13 sets of MFCC's in an audio group. What this produces is a set of 5 independent components for each of the 13 MFCC's in a group. These independent components are vectors representative of the driving temporal characteristics of that group of acoustic data at the frequency band represented by that Mel-frequency cepstral coefficient. The 5 independent components for each MFCC in a group get concatenated on the third dimension to produce a matrix containing the 13 points of comparison (groups of independent components) for that group. The points of comparison relative to the input data can be better visualized by figure 4.3 and 4.4.

The next section will display the results of running the experimental process described above and analyze its separation of independent temporal information for the test tones, musical genres, and spoken world languages and the effect of varying parameters within the process. This thesis has thus far developed a biologically inspired audio signal process to

extract the independent temporal features of acoustic information within different classes. The information extracted from this process is representative of the patterns and features for classes of acoustic data that demonstrate a significant amount of separation, so that it is later possible to extend this method to implement a computationally efficient method for classifying that information.

Chapter 5

Results Analysis

The previous two sections of this thesis outlined the reasoning and implementation of the experimental process used to demonstrate separation of acoustic data using independent component analysis. This section will outline the different types of experiments performed on each part of the data set, presenting and analyzing the results obtained from those tests at each step. In each applicable case, the tests will show that the different classes of acoustic information exhibit the statistical independence and temporal separation necessary to be classified. The results from running the test tones through this process will show what features of audio data that principal components and independent components can represent, further demonstrating how, in practice, ICA produces components with stronger properties components produced with PCA. The experiments done with musical genres and spoken world languages will show that the points of comparison are unique across different classes of acoustic information, justifying the conclusion that this process can sufficiently separate groups of audio data.

5.1 Initial Experiments (Tones)

The test tone experiments were much simpler then the full experiments done with musical genres and world languages. For test tones, a great deal of importance is in testing audio equipment's response at different frequencies, so many tones at different frequencies come in a set of test tones. However, I was concerned only with how PCA and ICA captured

the features of these tones, so as long as the frequency was constant in the tone sample then the magnitude of the frequency was irrelevant. Three types of tones provided a good idea of how PCA and ICA capture audio structure, bursts (saw tooth wave, 1KHz), pulse (square wave, 1KHz), and frequency ramp (20Hz-20KHz). Some properties of these test tones remained constant over all of the tests, mainly that there was only one significant principal component. The highest eigenvalue for the tones was always on the order of 10^5, whereas the rest of the non-zero eigenvalues were much less then 10^2. This indicates that the features of the simple tones can be entirely represented in one component, which is to be expected with such controlled sound.

The first tone processed was considered to be the simplest and most controlled sample of data, the square wave pulse. This was an artificially generated perfect square wave of constant frequency, constant amplitude, and instantaneous switching from positive to negative amplitude. When processed with MFCC's, quite obviously there was only sparsified values in one of the coefficients corresponding to 1KHz. It makes sense then that only one component would be extracted from this data, because all other coefficients are zero. This is exactly what happened and can be seen in figure 5.1. You can see from the plot and histogram of outputs that the principal component consists of one value almost entirely, representing the constant frequency of the tone.

The next test tone of interest is the 1KHz burst, which includes not just a constant frequency, but some intermittent silence to produce a constant repetitive beat over time. The pulse is also a saw tooth wave, so that there is no perfect transfer from positive to negative amplitude, but a gradual change. This will introduce some variance into the signal, because some windows of the MFCC process will no pick up part of the pulse, and some silence and calculate a coefficient for the mixed signal. In this case, you would expect to see a short range of values in the histogram between the peak principal component value and zero, when a window has captured nothing but silence. The plots in figure 5.2 show just that with the histogram comprising mostly of the peak value (all constant frequency) and zero (all silence), and a little bit of a middle value (some silence, and some frequency).

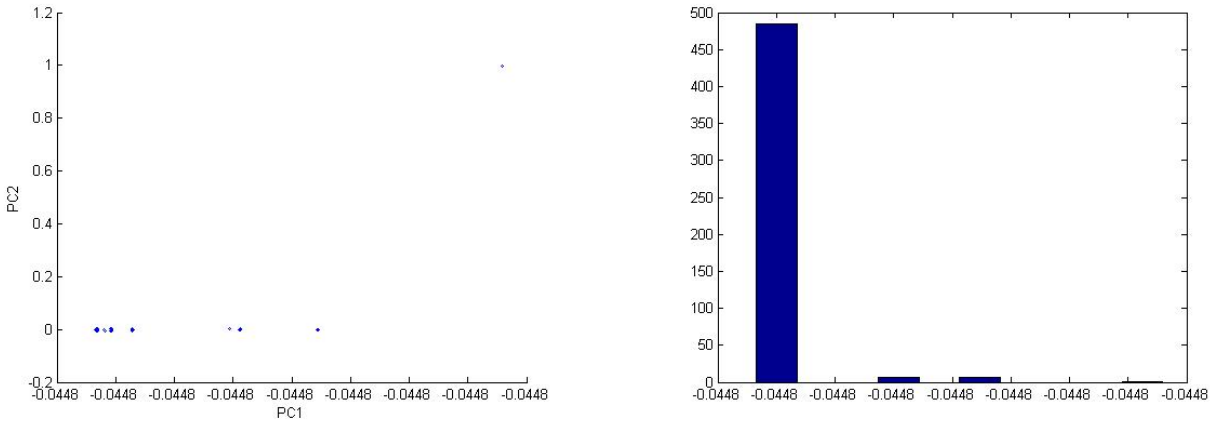

Figure 5.1: Square wave plot of principal components (left) and histogram of principal component 1

The last test tone is different in that it has a varying frequency over time. Because the interesting feature extracted from the previous tones has been some representation of their constant frequency, the frequency ramp tone should produce some plots of more interest. With the principal components maintaining some constant value for periods of constant frequency, one should expect to see a range of values in the component representing the range of frequencies in the tone. The interesting part of the test run with this tone is that there was still just one significant principal component, showing that PCA and ICA can represent data of varying frequencies quite easily. The plot of the principal component and of the magnitude of the component over time is shown in figure 5.3.

5.2 Musical Genres

After generating the results with the test tones from the previous section, it is reasonable to extend this process to fully realistic audio signals in the form of music. In this section, results will be presented for the experiment described in the **Process Implementation** chapter. The results collected will then be displayed and explained in the context of how

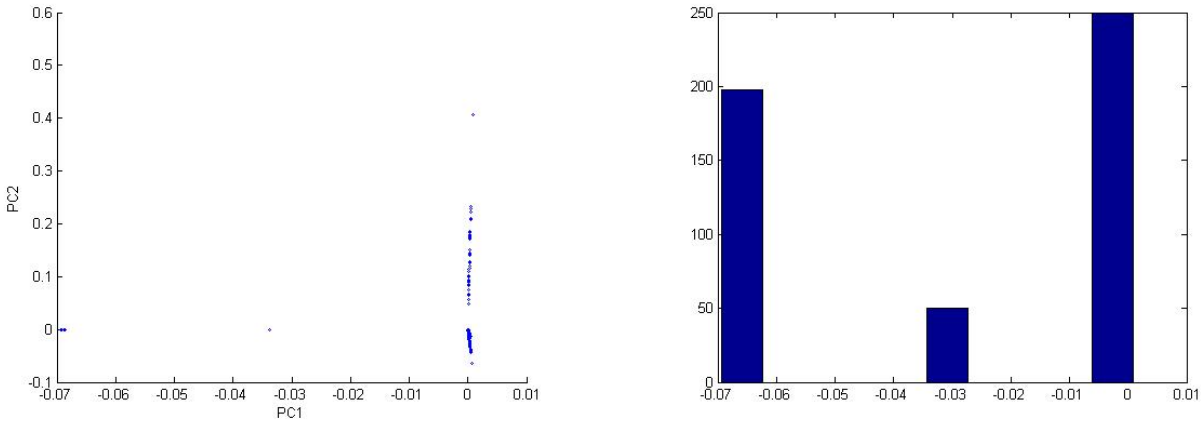

Figure 5.2: Burst sawtooth wave plot of principal components (left) and histogram of principal component 1

this process has given an efficient means of separating the independent temporal features of different musical genres. For the experiment described (40 songs, 10 songs per genre), the Matlab code used runs in approximately 21 seconds (average over 10 trials), a more then reasonable time to train the system on four genres of music. For the 13 sets of coefficients over 4 groups of data, the FastICA algorithm converged on the independent components in an average of 8 iterations, which is typical [14]. For all of the figures depicted in this section, the four genres will be color coded as Classical=Cyan, Jazz=Red, Metal=Green, and Electronic=Black.

5.2.1 Principle Component Separation

The separation of musical genres was preliminarily explored in [16] to demonstrate statistical independence across different genres of music, but while analyzing all of the cepstral coefficients for a single sample at once. This process derives components for each Mel-frequency cepstral coefficient for multiple samples separately. In order to demonstrate that this process exhibits the same characteristics of statistical independence, experiments were

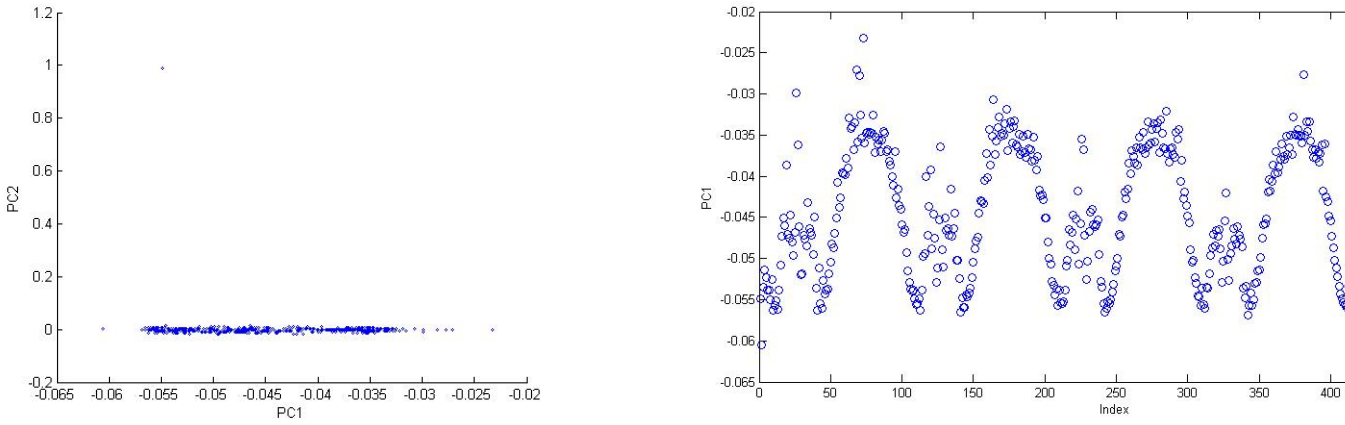

Figure 5.3: Frequency ramp plot of principal components (left) and principal component 1 over time

run on the musical genre database with principal component analysis inserted for ICA. Figure 5.4 shows plots of latent dimensions extracted using PCA for several Mel-frequency cepstral coefficients that depict the relationship of the principal components for each genre relative to each other at that Mel-frequency cepstral coefficient.

As you can see from the various graphs in figure 5.4, the principal components show signs of statistical independence across a number of Mel-frequency cepstral coefficients. Similar results were yielded for all of the coefficients, and using more then just the first two principal components. The musical genres produced between 5 and 10 significant principal components for each coefficient, which could all be used as points of comparison in separating out the different genres, or matching new acoustic information to a particular genre. The significance of the component is measured by its eigenvalue, which for these tests were on the order of 10^2 or higher for significant components. Even though the principal components are ranked by their eigenvalues, and therefore significance, the best separation of acoustic classes may be in the third, fourth, etc. significant component. This is why in figure 5.4 the first and third components are displayed. One can gain a better idea of what the separation across more coefficients would be like by the three dimensional graphs

in figure 5.5. These plots give a good idea of the different spaces each genre occupies in higher dimensions.

For principal component analysis, the ray structure of the different genres shows statistical independence. Each ray, or tight grouping of signals represents some period in the sample data of variance, either in frequency, amplitude, or both, like a complex combination of the characteristic plots seen with the test tones. These collection of variances in the sample data cause the principal components to occupy some arrangement of high-dimensional space at each Mel-frequency cepstral coefficient, which across all of the coefficients could be quite unique for each genre. What independent component analysis can do with this data is draw out the underlying temporal themes that account for a great deal of this variance and patterns in the sample data. The next section explores the results of using this process with ICA.

5.2.2 Independent Component Separation

This extension to the work presented above uses independent component analysis, a much stronger statistical transform [1], to extract out the independent temporal features and patterns of music samples that will demonstrate the greatest separation amongst different genres. The process for this has been outlined earlier in this thesis and has done a good job of producing the desired results. The first point to be made is that the differences in independent components of each genre exist across all of the Mel-frequency cepstral coefficients. These differences are highlighted in figure 5.6 where a couple of the independent components are plotted against each other over a number of the MFCC's to demonstrate the uniqueness of genres.

As can be seen from the 2-dimensional graphs in figure 5.6, the independent temporal structures of the musical genre data has been estimated and formed orthogonal rays of data points. Because all of the temporal components are independent of one another, when plotted one against the other, most of their data points will lie on the axis. This makes it hard to display unique structure and shape on just a 2-dimensional graph. However, one

notable point to make is that the data *is* much more structured then the plots of the principal components. Also, because all of the components are independent and unordered, no one independent feature accounts for any more variance or themes in the sample music then another. Therefore, all of the extracted independent components in all of the genres have to be taken into account in order to display uniqueness across different genres of music. This high dimensional separation of independent components can be better seen in the plots in figure 5.7, although it is still limited to only displaying 3 dimensions.

The plots in figure 5.7 give a multi-dimension depiction of the types of patterns the independent components extracted by this process produce. The problem with separating temporal features extracted by independent component analysis is due to the inherent ambiguities in the ICA process. Independent components are ambiguous with regards to amplitude, phase, and order. Some of these ambiguities can be corrected for by normalizing the data, such as amplitude and phase. Determining uniqueness is not as simple as seeing the plotted points occupy different space on the graphs, but it can be partially illustrated by them. The separation of musical genres using independent component analysis can be determined by the structure of all of the extracted temporal features, for all of the Mel-frequency cepstral coefficient frequency ranges. Each of these features represents some aspect of the original sample song within that Mel-frequency cepstral coefficient, such as repetitive sound like bass drum hits, or a progressive change in frequency or key of the rhythm section.

This process uses several different and independent temporal features for each Mel-frequency cepstral coefficient to provide an all-encompassing signature for a particular class of acoustic information. Accounting for many underlying themes in the audio data at each MFCC frequency range means that temporal features will be extracted from each one that are shared with some features from other classes, and that some extracted features will be unique to that genre. The combination of both is still unique as the components one class might share with another is still a distinct characteristic of that class. The graphs in figures 5.6 and 5.7 give a good representation of the shared and unique characteristic

independent features of the four genres used in this experiment. In the next section, this characteristic separation of genres of music will be extended to spoken world languages to study the similarities and differences in the groupings of the two forms of acoustic data.

5.3 World Language

In this section, the process used to obtain the characteristic temporal feature extraction for musical genres will be applied to the spoken world language database described earlier in some basic experiments. The results in this section will correlate back to the results obtained in the musical genre section to compare and contrast the temporal feature groupings of musical genres versus the temporal feature groupings of people speaking different languages. The experiment for this section will be run with two different languages (English and Hindi), with 5 speakers in each, to determine the uniqueness of the features in each language.

5.3.1 Principle Component Separation

As with the musical genres, an experiment was set up using principal component analysis initially to generate latent dimensions for the speech samples that will exhibit signs of statistical independence, structure, and separability. This experiment was set up with the two languages of English and Hindi, cyan and red respectively in all of the plots in this section. These two languages are quite different and easily distinguishable to any person listening to them. However, in audio processing, and especially speech processing, there are inherent problems in comparing the context of speech and its higher level features because all humans have similar physical processes that they speak with. Human speech itself can be closely related enough to be classified apart from other forms of acoustic information, so to distinguish between types of speech is not trivial. The high level temporal feature extraction done by the process developed in this work, offers a good solution to this problem as world languages vary greatly in their temporal structure.

For instance, the frequency and nature of certain sounds, pauses, syllables, and transitions over time are the characteristics that make languages different. By extracting underlying temporal features from speech data, it is reasonable to propose that those characteristics of world languages will be well represented and distinguishable amongst other world languages. The plots shown in figures 5.8 and 5.9 illustrate this separation of temporal characteristics well.

Just as in the principal component analysis of the music genres, the same ray structure and separation of features is apparent in the plots of world languages in figure 5.8. These principal features of the world languages however are accounting for variance in the sample signal, which are simple features like syllables and speaking frequency. Independent component analysis will be able to extract out the temporal features that better represent the context over time of the speech. This is useful, and especially necessary for speech processing as similar principal features like particular sounds and frequencies will span multiple languages, but temporal features that take into account more characteristics that distinguish languages will provide better separation of world language data.

5.3.2 Independent Component Separation

This section will explore the results obtained from performing the experiment done with spoken world languages and PCA above with independent component analysis. The significance of independent component analysis and what yields the interesting results from this process is its ability to extract independent temporal themes and patterns from data. In the case of spoken languages, ICA should draw out higher-level features of the speech that relate to the delivery, transitions, pauses, and flow of the language over time. This information is important because it's these characteristics that make different languages unique. The graphs of independent component data from the 2 language experiment in figures 5.10 and 5.11, show that separation of unique characteristics between English and Hindi.

In the plotted language data, it's quite obvious that there exists statistical separation

between English and Hindi. Similar to the data accumulated for the music genre classification, there are sometimes shared characteristic independent components, as well as unique ones. The combination of those components and across all of the Mel-frequency cepstral coefficients produces a unique set of temporal features for the varying frequency ranges in an audio sample. So information for separation and classification is being take into account in both the time and frequency domain, allowing for a comprehensive method of representing classes of acoustic information. The separability of the music genres and spoken world languages, two very different forms of acoustic information, is sufficient proof that this work the basis for a general method of acoustic separation and classification.

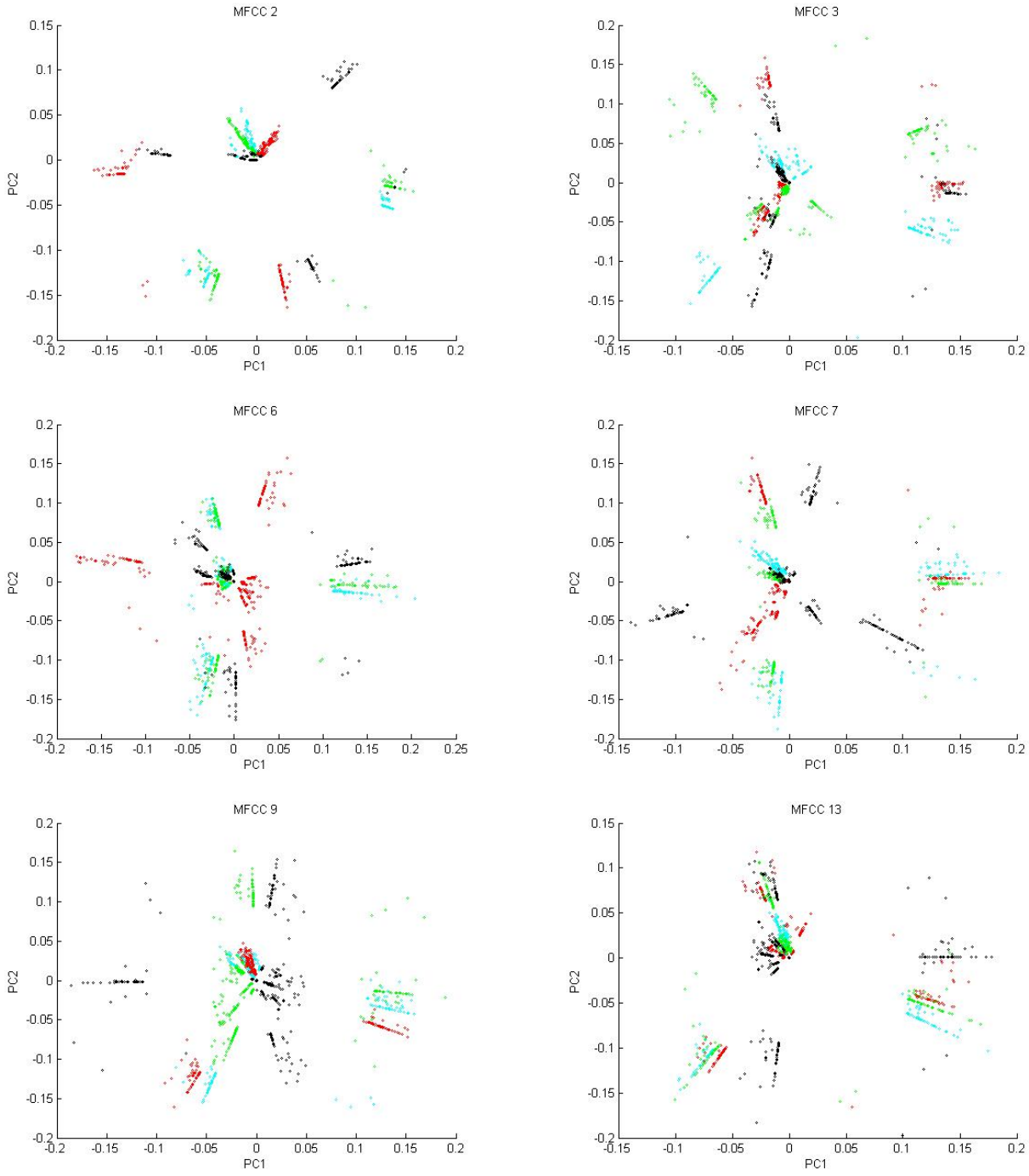

Figure 5.4: Principle components for 6 different MFCC's, demonstrating statistical independence and separation (2D)

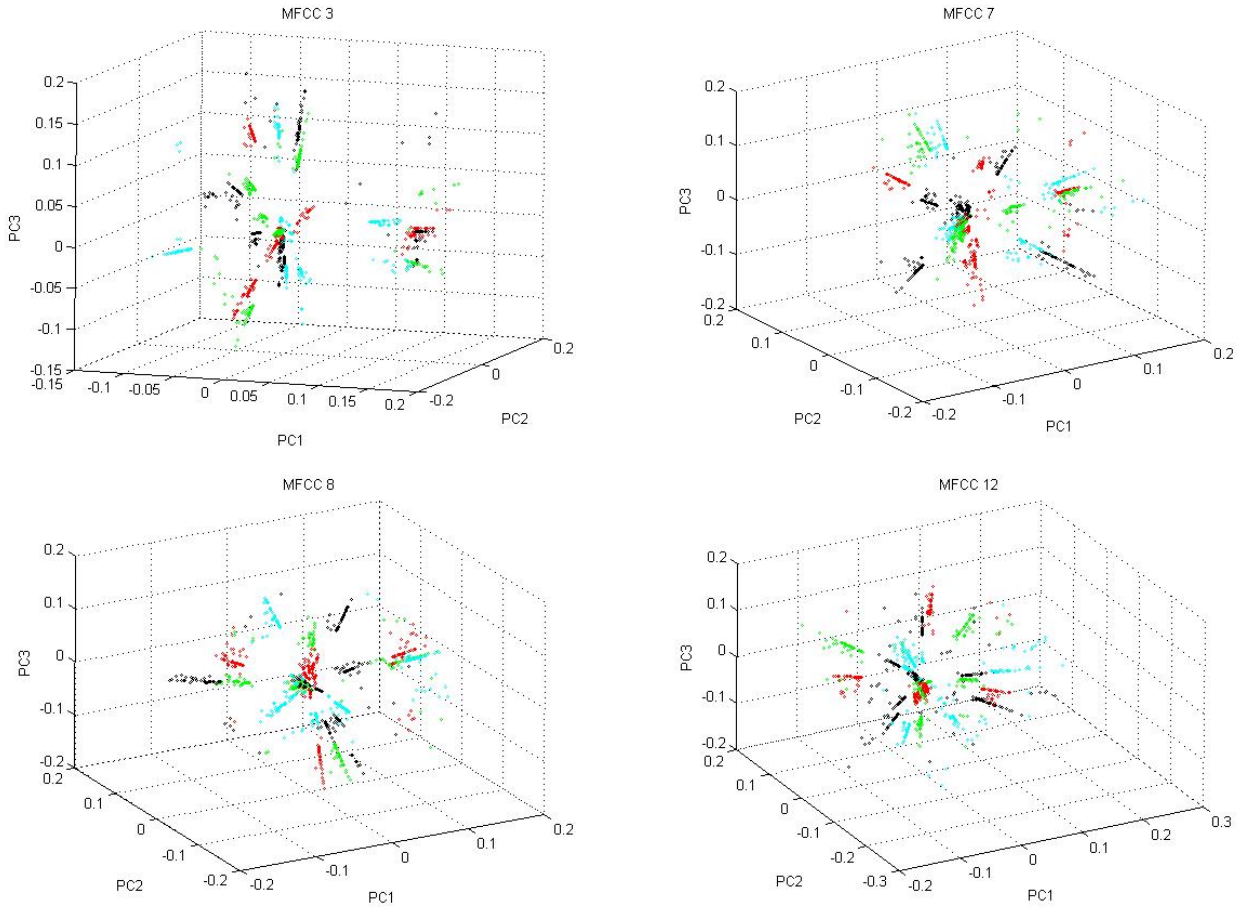

Figure 5.5: Principle components for 4 different MFCC's, demonstrating statistical independence and separation (3D)

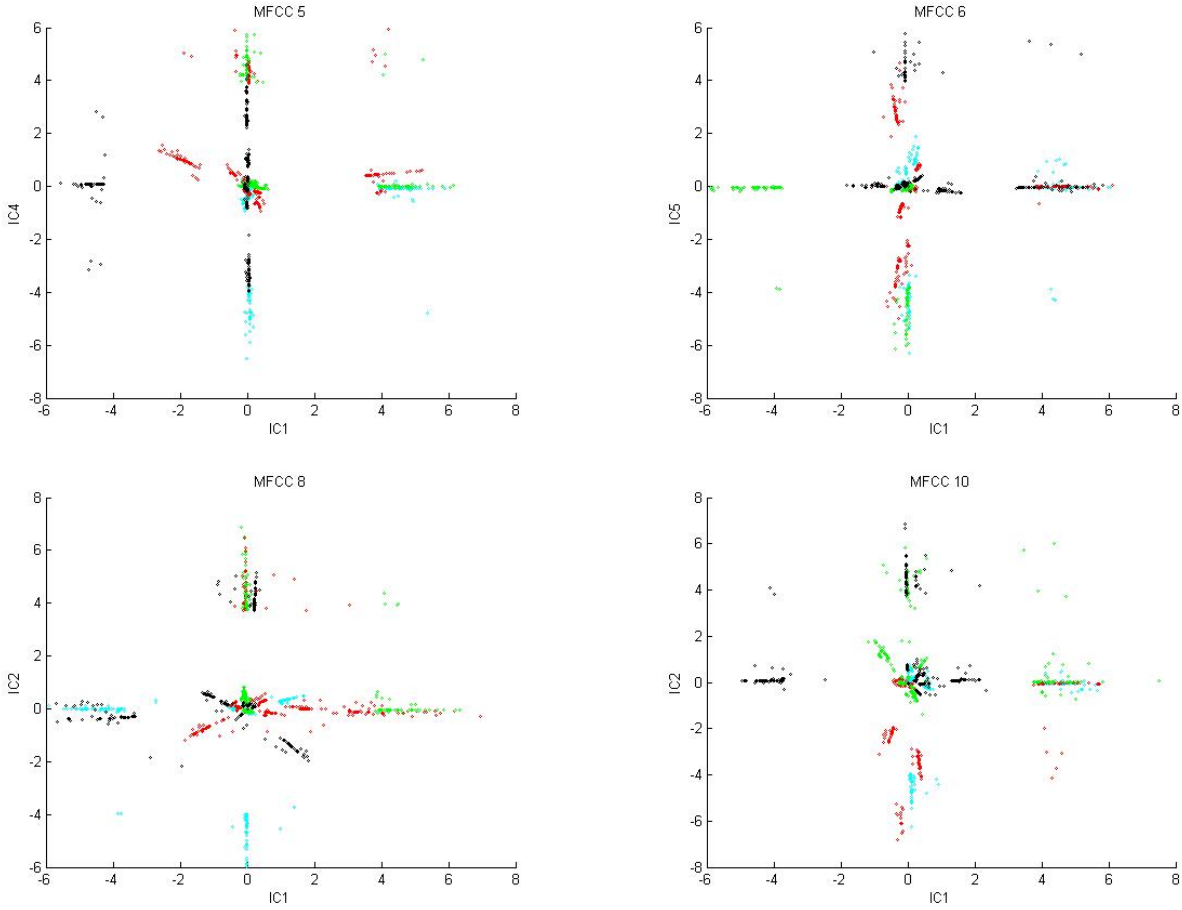

Figure 5.6: Independent temporal features for 4 different MFCC's (2D)

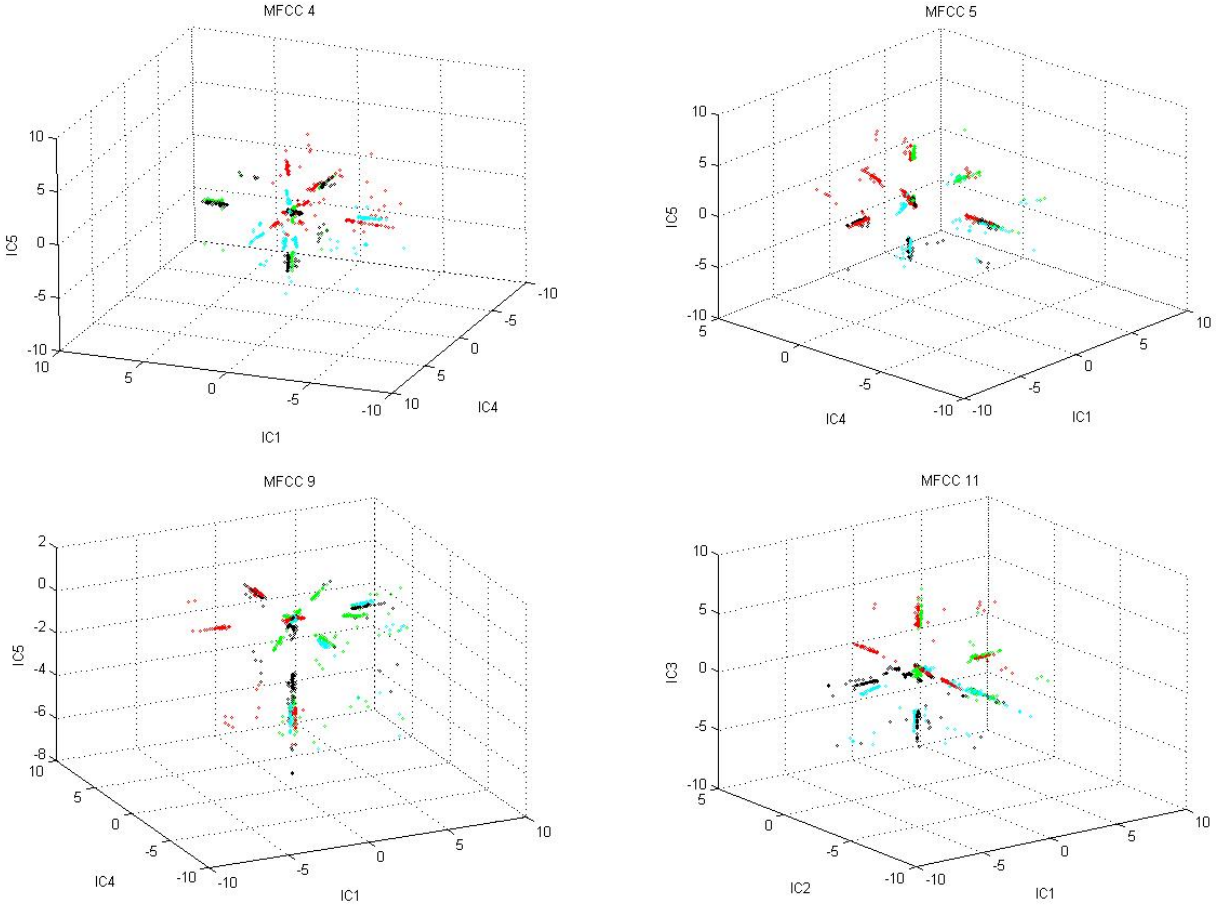

Figure 5.7: Independent temporal features for 4 different MFCC's (3D)

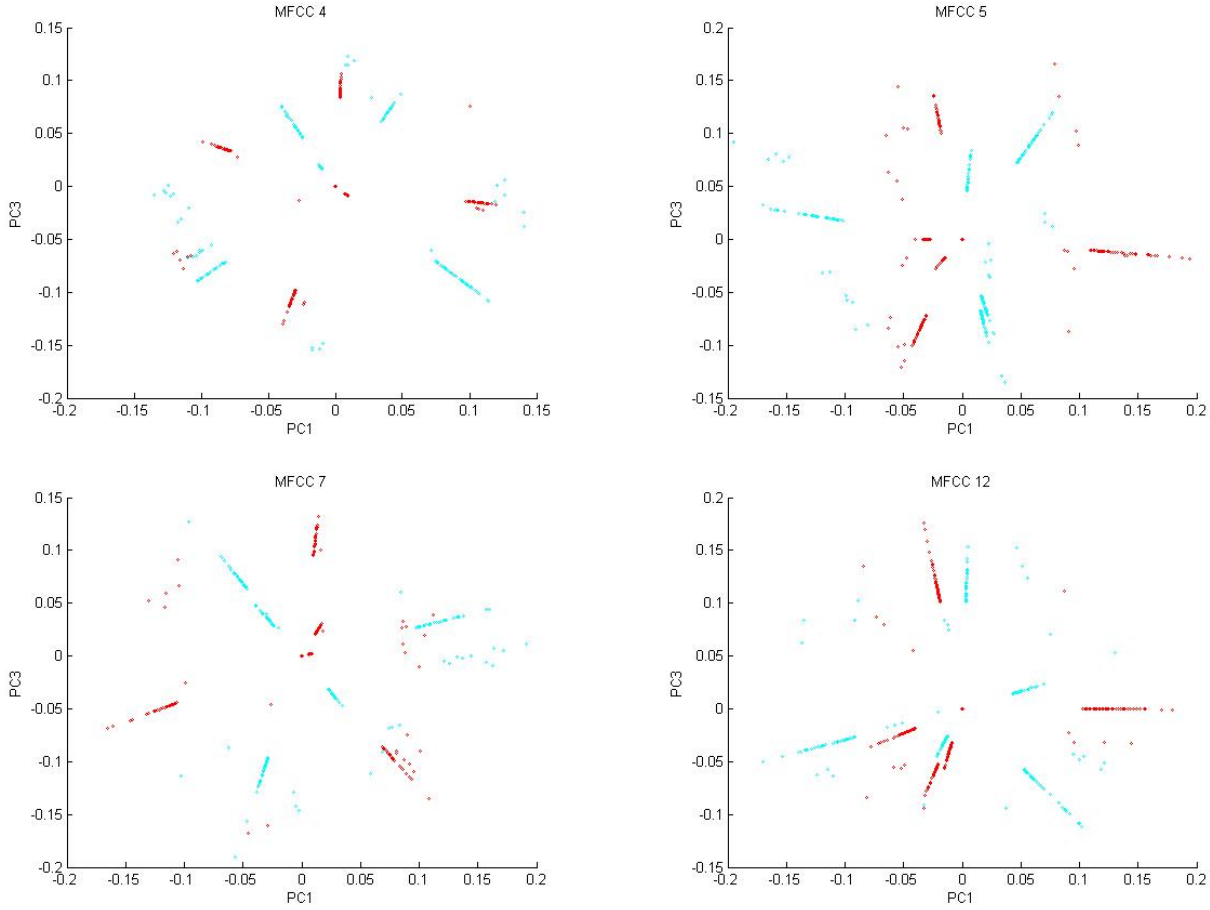

Figure 5.8: Principle components for 4 different MFCC's, demonstrating statistical independence and separation (2D)

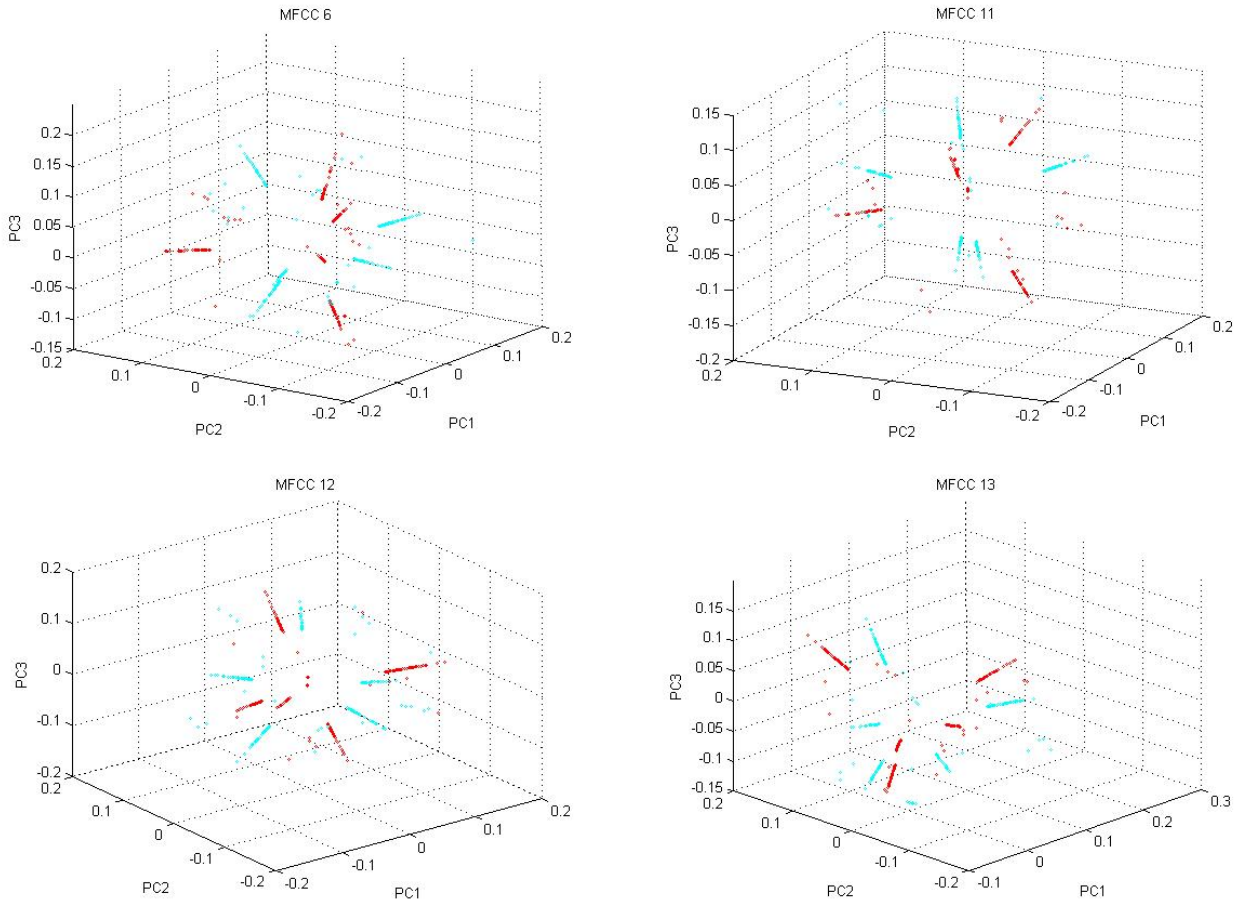

Figure 5.9: Principle components for 4 different MFCC's, demonstrating statistical independence and separation (3D)

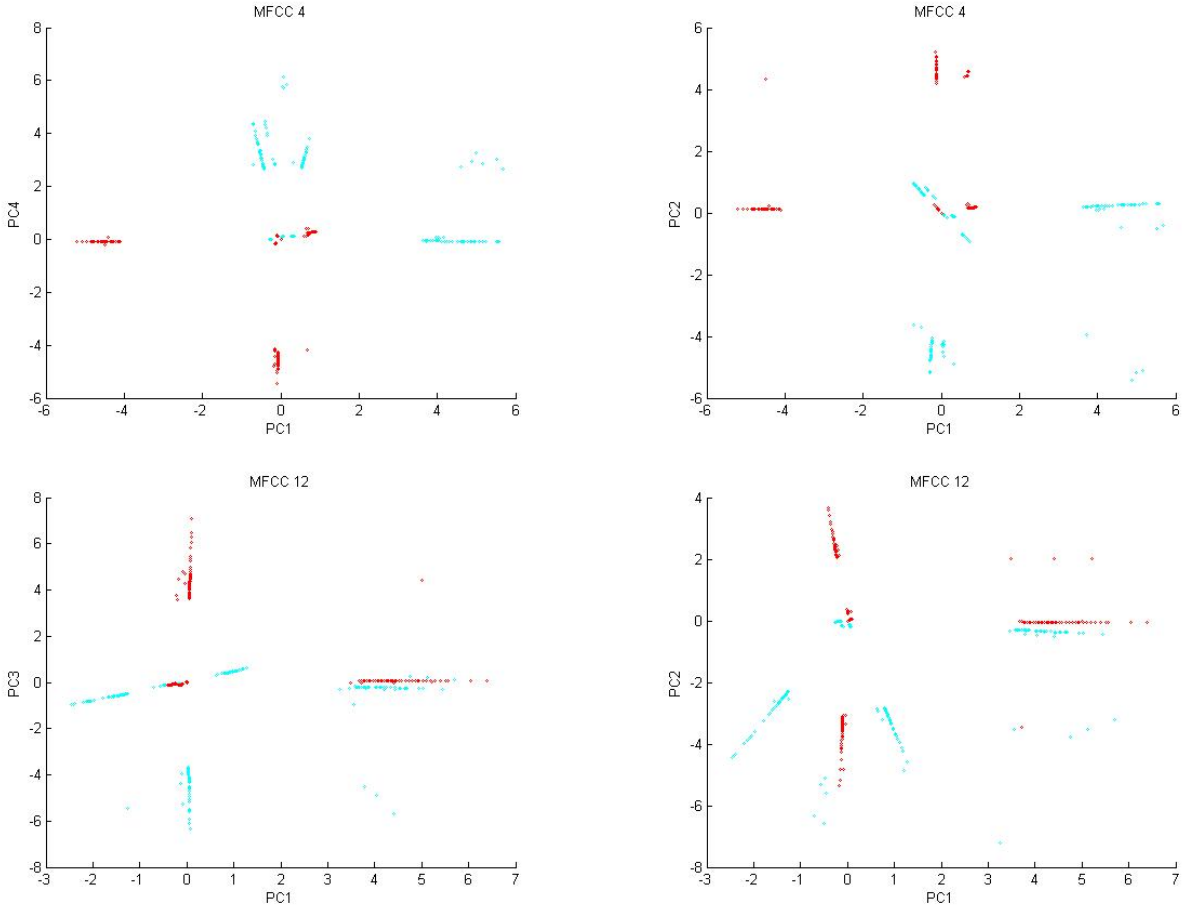

Figure 5.10: Independent components for 4 different MFCC's (2D)

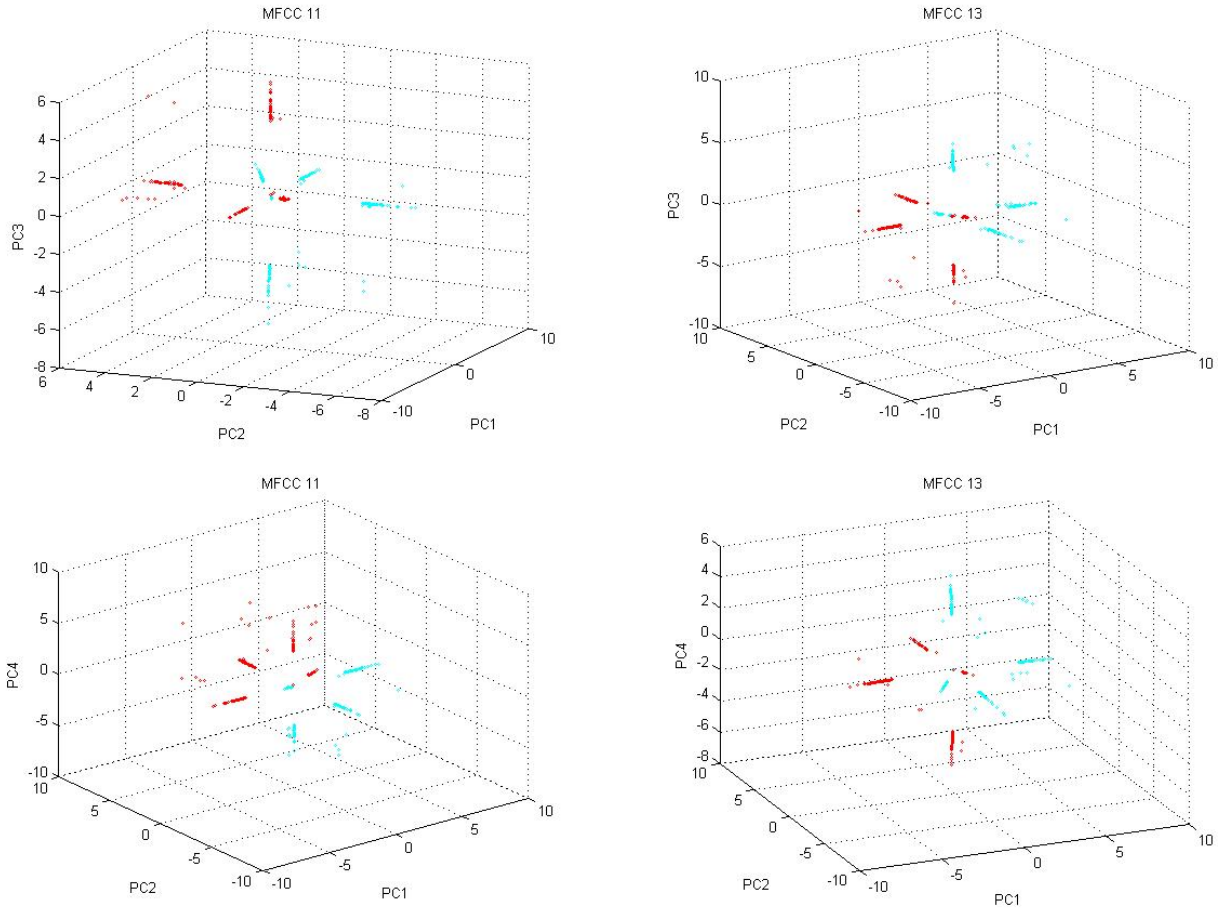

Figure 5.11: Independent components for 4 different MFCC's (3D)

Chapter 6

Conclusions

This experimental work has successfully demonstrated that through the preprocessing of Me-frequency cepstral coefficients and underlying temporal feature extraction of independent component analysis, a computationally efficient process for acoustic data separation and classification can be developed. The cognitive model for this process was constructed from previously known works and novel thought and experimentation in the area of independent component analysis and the characteristics of high-dimensional data. This work was then extended from the common area of musical genre classification to separating the characteristics of world languages with the same process, therefore proving a general method for acoustic classification based on the biological processing of the human auditory cortex, and current areas of cognitive modeling.

Chapter 7

Future Work

This thesis has provided a process for a computationally efficient way to extract the temporal features of acoustic information and demonstrate their tendency towards separation in high dimensional space. With the separation of classes of acoustic data, the next intuitive step is to classify those separated classes. Because the uniqueness of the data extracted and observed here exists in a number of dimensions and a number of points of comparisons (1 for each Mel-frequency cepstral coefficient), a high-dimensional data classifier could be employed to provide an unsupervised learning environment that could separate and cluster classes of acoustic data without prior grouping. Advanced clustering algorithms and self-organized maps lend themselves to this type of classification and grouping. Another area of extension of this work could be into the classification of new acoustic information introduced to the system trained on the known classes of audio data used in this thesis. This application of dynamic classification would allow the process to easily be updated with new additions of audio files to the database, and consequently update the underlying temporal themes that represent that class within the database. This would enable the process to be completely dynamic and adapt to changes in the acoustic database.

Bibliography

[1]

[2] Auditory System: Anatomic Tour. Available at http://serous.med.buffalo.edu/-hearing/auditory_cortex.html.

[3] FastICA website. http://www.cis.hut.fi/projects/ica/fastica/.

[4] Wikipedia. http://www.wikipedia.org.

[5] P. Agarwal and M. Kukreja. Text Independent Speaker Recognition. Personal Publication (Available at http://www.cse.iitd.ernet.in/c̃s1030178/mysite2/html), 2005.

[6] Karthik Chandrasekaran. Independent Component Analysis Presentation, 2005.

[7] Eric Choi. On Compensating the Mel-Frequency Cepstral Coefficients for Noisy Speech Recognition, 2006.

[8] Dan Ellis. Audio Processing Toolbox. http://www.ee.columbia.edu/ dpwe/resources/matlab/mp3read.htr

[9] L. Hansen and L. Feng. On Low-level Cognitive Components of Speech. STVF No. 26-04-0092, April 2005.

[10] Peter J. Huber. Projection Pursuit. In *The Annals of Statistics*, page 435, June 1985.

[11] Edgar Berdahl Hyoung-Gook Kim and Thomas Sikora. Study of MPEG-7 Sound Classification and Retrieval. Communication Systems Group, Technical University of Berlin, Germany.

[12] A. Hyvarinen and U. Koster. FastISA: A Fast Fixed-Point Algorithm for Independent Subspace Analysis. Personal Publication (Available at http://www.cs.helsinki.fi/u/-koster/ESANN2006.pdf), 2000.

[13] A. Hyvarinen and E. Oja. A Fast Fixed-Point Algorithm for Independent Component Analysis. Neural Computation, 1997.

[14] A. Hyvarinen and E. Oja. Independent Component Analysis: Algorithms and Applications. Neural Networks, 13(4-5), 2000.

[15] Aapo Hyvarinen. Survey of Independent Component Analysis. In *Neural Computing Surveys 2*, page 94, 1999.

[16] P. Ahrendt L. Hansen and J. Larsen. Towards Cognitive Component Analysis. Personal Publication (Available at http://www2.imm.dtu.dk/pubdb/views/-edoc_download.php/3591/% -pdf/imm3591.pdf).

[17] J. Lindgren and Aapo Hyvarinen. Learning High-Level Independent Components of Images Through a Spectral Representation. In *International Conference Proceedings on Pattern Recognition (ICPR2004)*, 2004.

[18] Beth Logan. Mel-Frequency Cepstral Coefficients for Music Modeling. Published by Compaq Computer Corporation (Available at http://ciir.cs.umass.edu/music2000/-papers/logan_abs.pdf), 2000.

[19] J. Larsen M. Pedersen, T. Lehn-Schioler. BLUES from Music: BLind Underdetermined Extraction of Sources from Music.

[20] J. Larsen P. Ahrendt and A. Meng. Decision Time Horizon for Music Genre Classification Using Short Term Features. Personal Publication (Available at http://eprints.-pascal-network.org/archive/00000154/01/eusipco04rev2.pdf), 2004.

[21] M. Hamalainen R. Hari R. Vigario, V. Jousmaki and E. Oja. Independent Component Analysis for Identification of Artifacts in Magnetoencephalographic Recordings. In *Advances in Neural Information Processing Systems*, 1998.

[22] L. R. Rabiner and B. H. Jueng. Fundamentals of Speech Recognition. Prentice Hall, 1993.

[23] Josef Rauschecker and Biao Tian. Mechanisms and Streams for Processing "What" and "Where" in Auditory Cortex. In *Proceedings of the National Academy of Sciences*, page 11800.

[24] S. Ravindran S. Kamath and D. Anderson. Independent Component Analysis for Audio Classification. In *IEEE 11th Digital Processing Workshop*, February 2004.

[25] E. Sahouria and A. Zakhor. Content Analysis of Video using Principal Components. In *1998 International Conference on Image Processing*, pages 541–545, October 1998.

[26] Lindsay I. Smith. A Tutorial on Principle Component Analysis. Personal Publication (Available at http://csnet.otago.ac.nz/cosc453/student_tutorials/-principal_components.pdf), February 2002.

[27] Matthew Turk and Alex Pentland. Eigenfaces for Recognition. Personal Publication (Available at http://www.cs.ucsb.edu/m̃turk/Papers/mturk-CVPR91.pdf), 1991.

Appendix A

Source Code (MATLAB)

The following fully commented Matlab code files are the sources used to generate the results presented in this thesis based on the database described.

A.1 README.m

```
% README.m
%
% Author:  James L. Brock
% Date:  May 8, 2006

%%%%% Overview %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This directory contains the source code for my thesis, which explored the
% feasibility of using Mel-frequency cepstral coefficients and independent
% component analysis to derive temporal features of music genres and world
% languages that demonstrated a good separation of temporal features.  These
% features could possibly be used for classification in future work.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Table of Contents %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Chapter 1:  File List
% Chapter 2:  File Descriptions
% Chapter 3:  Miscellaneous Remarks
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Chapter 1:  File List %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1) init.m
% 2) main.m
% 3) output.txt
% 4) plotComps3.m
% 5) plotComps.m
% 6) princompN.m
% 7) README.m
% 8) readNextFile.m
% 9) soundtest.m
% 10) sparsMFCC.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Chapter 2:  File Descriptions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1) init.m
% This function returns a list of files in the specified directory.  The
% function assumes that all file names are of the same length.
%
% 2) main.m
% This script will execute the experiment specified by the global
% variables defined at the beginning of the script, producing A LOT of
% 2-D and 3-D plots of the principal or independent components of the
% sample audio files in the database.
%
% 3) output.txt
% Example output debug file
%
% 4) plotComps3.m
% This function produces 3-D plots of the specified components in
% trainDataOut for a certain MFCC across 4 genres.
%
% 5) plotComps.m
% This function produces 2-D plots of the specified components in
% trainDataOut for a certain MFCC across 4 genres.
%
```

```
% 6) princompN.m
% a copy of the princomp.m file included in the statistics toolbox for
% Matlab that was moved to my home directory and renamed to avoid
% errors.
%
% 7) README.m
% This file!
%
% 8) readNextFile.m
% This file reads the specified file with the specified parameters
% using the mp3read function.
%
% 9) soundtest.m
% This script creates data of principle and independent components of
% test tones.
%
% 10) sparsMFCC.m
% This finction creates a matrix of sparsified MFCC's for the specified
% file.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Chapter 3:  Miscellaneous Remarks %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1) See any of the source files for more information on parameters and
% useage
%
% 2) Refer to my thesis writeup and primary resources for why certain
% variables are set to the values they are.
%
% 3) Thesis website:  http://www.livefreeordienh.com/thesis
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.2 main.m

```
% main.m
%
% Author:  James Brock
% Date:  April 24, 2006
% Description:  This script will execute the experiment specified by the
% global variables defined at the beginning of the script, producing A LOT
% of 2-D and 3-D plots of the principal or independent components of the
% sample audio files in the database.

% Clean up the Matlab workspace and suppress warnings.  Warnings are
% suppressed because the mfcc function throws A TON of them and they're
% negligible.
clear;
clc;
warning 'off';

% Global Variables and constants
TRAINDIR = './db/train/'; % directory containing mp3 files
OUTFILE = 'output.txt'; % debug output file name
```

```
trainFiles = 40; % number of files to use
nIC = 5; % number of independent components
% to keep
thr = 0.95; % threshold for sparsifying MFCC's
groups = 4; % number of acoustic groups
coefs = 13; % number of MFCC's calculated for
% each file (don't change!)
channels = 1; % number of audio channels to read
downSamp = 2; % downsampling rate
samples = [50000 (50000+220500)]; % sample space in each mp3 file to
% read

% Start the timer ...  see how fast it runs!
tic;
% Create the list of file names to use in this experiment
trainList = init(TRAINDIR, trainFiles);
% Initialize output log file
out = fopen(OUTFILE, 'w');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Run the experiment* %
% *Assumes an equal number of training songs per group %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
trainDataIn = []; % ALL Sparsified MFCC's
plane = []; % Sparsified MFCC's for a particular group
fPerG = (trainFiles/groups); % Training files per group

% Calculate the sparsified Mel-Frequency cepstral coefficients for each
% training sample in each genre.
for i=1:trainFiles
groupNum = floor((i-1)/fPerG);
fileName = trainList(i,:);
fprintf(out, 'FILE: %s \n', fileName);
[file, rate] = readNextFile(fileName, channels, downSamp, samples);
lowFeat = sparsMFCC(file, rate, thr);
tempFeat = [];
for j=1:coefs
tempFeat = cat(3, tempFeat, lowFeat(1,:));
end
% Concatenate MFCC's for each group
plane = cat(1, plane, tempFeat);
% When a group completes processing, concatenate that groups MFCC's
% into one 4-D matrix of all data
if ( mod(i,fPerG) == 0 )
trainDataIn = cat(4, trainDataIn, plane);
fprintf(out, 'Current File:  %s, Current Index:  %i\n', fileName, i );
plane = [];
end
end

% Calculate the components for each MFCC of each genre
trainDataOut = [];
coefData = [];
```

```
for i=1:groups
for j=1:coefs
if (trainDataIn(:,:,j,groups) == 0)
plane = trainDataIn(:,:,j,groups);
else
% Calculate independent components (comment out for PCA)
plane = fastICA(trainDataIn(:,:,j,groups), 'approach', 'symm', ...
'g', 'tanh', 'lastEig', (fPerG/2), 'numOfIC', nIC, ...
'displayMode', 'off');
% Calculate principal components (comment out for ICA)
%[plane, j2scores, j2latent, j2tsq] = princompN(trainDataIn(:,:,j,i));
%plane = plane(:,1:(fPerG/2))';
end
coefData = cat(3, coefData, plane);
end
fprintf(out, '%i %i %i \n', size(coefData)); %for debug only
trainDataOut = cat(4, trainDataOut, coefData);
coefData = [];
if ( size(size(trainDataOut)) == [1 3])
fprintf(out, 'Group:  %i, Data Size:  %i %i %i %i \n', i, ...
size(trainDataOut), 1 ); %for debug only
else
fprintf(out, 'Group:  %i, Data Size:  %i %i %i %i \n', i, ...
size(trainDataOut) ); %for debug only
end
end

%%%%% Plot some of the data %%%%%
% 2D Plots
for i=1:coefs
plotComps(trainDataOut, i, 1, 2);
end
for i=1:coefs
plotComps(trainDataOut, i, 1, 3);
end
for i=1:coefs
plotComps(trainDataOut, i, 1, 4);
end
for i=1:coefs
plotComps(trainDataOut, i, 1, 5);
end

% 3D Plots
for i=1:coefs
plotComps3(trainDataOut, i, 1, 2, 3);
end
for i=1:coefs
plotComps3(trainDataOut, i, 1, 2, 4);
end

% Stop timer and save time difference in a variable (runTime)
runTime = toc;
% Completed
```

```
fprintf(out, 'EXPERIMENT COMPLETED!\n');
% Close debug output file
fclose(out);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.3 init.m

```
% init.m
%
% Author:  James Brock
% Date:  April 24, 2006
% Description:  This function returns a list of files in the specified
% directory.  The function assumes that all file names are of the same
% length.
%
% Parameters:
% dbdir - directory to retrieve file names from
% maxImg - the maximum number of file names to return
%
function [fileList] = init( dbdir, maxImg )
% Generate list of files in the directory
files = dir(dbdir);
fileNum = size(files,1);
% Shave off the .  and ..  directory listings
files = files(3:fileNum,:);
fileNum = fileNum - 2;
if (maxImg > fileNum)
maxImg = fileNum;
end

% Comment the second line below it to return a randomized list of file
% names.
index = randperm(fileNum);
index = sort(index);
index = index(1:maxImg);

% Produce the vector of file name strings
fileList = [];
for i=1:maxImg
name = sprintf('%s%s', dbdir, files(index(i)).name);
fileList = cat(1, fileList, name);
end
end
```

A.4 readNextFile.m

```
% readNextFile.m
%
% Author:  James L. Brock
% Date:  April 24, 2006
% Description:  This file reads the specified file with the specified
```

```
% parameters using the mp3read function.
%
% Parameters:
% file - file name
% channels - 0 for mono signal, nonzero for stereo signal
% downSamp - downSampling factor, only 2 or 4 accepted
% samples - 2 element vector specifying the start and end point in the
% file
%
function [newFile, rate] = readNextFile(file, channels, downSamp, samples)
% Read next mp3 file in and return the file information
% sampling rate
[newFile, rate] = mp3read(file, samples, channels, downSamp);
end
```

A.5 sparsMFCC.m

```
% sparsMFCC.m
%
% Author:  James Brock
% Date:  April 24, 2006
% Description:  This finction creates a matrix of sparsified MFCC's for the
% specified file.
%
% Parameters:
% file - file data vector(1 channel)
% rate - sampling rate of the file data
% thr - threshold at which to keep MFCC's (0 to 1)
%
function [features] = sparsMFCC(file, rate, thr)
% Calculate Mel-frequency cepstral coefficients for the file
mels = abs(mfcc(file, rate));

% Determine top 5% magnitude threshold vallue
thresh = quantile(mels', thr);
%thresh = quantile(mels, thr);
features = mels;
sz = size(features);

% Sparsify the MFCC's to be zero unless in the top 5%
for i=1:sz(1)
for j=1:sz(2)
if (features(i,j) < thresh(i))
features(i,j) = 0;
end
end
end
end
```

A.6 plotComps.m

```
% plotComps.m
%
% Author:  James L. Brock
% Date:  April 30, 2006
%
% Description:  This function produces 2-D plots of the specified components
% in trainDataOut for a certain MFCC across 4 genres.
%
% Parameters:
% trainDataOut - 4-D matrix of #ofMFCC x MFCC length x samples x groups
% mfccX - the MFCC to target for these plots
% comp1,comp2 - components to plot against each other
%
function


= plotComps(trainDataOut, mfccX, comp1, comp2)
Xaxe = sprintf('PC%i',comp1);
Yaxe = sprintf('PC%i',comp2);
titleX = sprintf('MFCC %i', mfccX);
figure,
scatter(trainDataOut(comp1,:,mfccX,1),trainDataOut(comp2,:,mfccX,1),3,'c'),
hold,
scatter(trainDataOut(comp1,:,mfccX,2),trainDataOut(comp2,:,mfccX,2),3,'r'),
scatter(trainDataOut(comp1,:,mfccX,3),trainDataOut(comp2,:,mfccX,3),3,'g'),
scatter(trainDataOut(comp1,:,mfccX,4),trainDataOut(comp2,:,mfccX,4),3,'k'),
xlabel(Xaxe),ylabel(Yaxe),title(titleX);

end
```

A.7 plotComps3.m

```
% plotComps3.m
%
% Author:  James L. Brock
% Date:  April 30, 2006
%
% Description:  This function produces 3-D plots of the specified components
% in trainDataOut for a certain MFCC across 4 genres.
%
% Parameters:
% trainDataOut - 4-D matrix of #ofMFCC x MFCC length x samples x groups
% mfccX - the MFCC to target for these plots
% comp1,comp2,comp3 - components to plot against each other
%
function [] = plotComps3(trainDataOut, mfccX, comp1, comp2, comp3)
Xaxe = sprintf('PC%i',comp1);
Yaxe = sprintf('PC%i',comp2);
Zaxe = sprintf('PC%i',comp3);
titleX = sprintf('MFCC %i', mfccX);
figure,
```

```
scatter3(trainDataOut(comp1,:,mfccX,1),trainDataOut(comp2,:,mfccX,1),...
trainDataOut(comp3,:,mfccX,1),3,'c'),
hold,
scatter3(trainDataOut(comp1,:,mfccX,2),trainDataOut(comp2,:,mfccX,2),...
trainDataOut(comp3,:,mfccX,2),3,'r'),
scatter3(trainDataOut(comp1,:,mfccX,3),trainDataOut(comp2,:,mfccX,3),...
trainDataOut(comp3,:,mfccX,3),3,'g'),
scatter3(trainDataOut(comp1,:,mfccX,4),trainDataOut(comp2,:,mfccX,4),...
trainDataOut(comp3,:,mfccX,4),3,'k'),
xlabel(Xaxe),ylabel(Yaxe),zlabel(Zaxe),title(titleX);

end
```

A.8 soundtest.m

```
% soundtest.m
%
% Author:  James L. Brock
% Date:  April 1, 2006
% Description:  This script creates data of principle and independent
% components of test tones.
%
clear
clc

%Set parameters
w = 0.030; %sample window size in sec
v = 0.010; %sample window overlap in sec
k = 13; %number of MFCC's to use
thresh = 0.90;
%test = 'freqsaw/s20-20k.mp3';
%test = 'misctests/burst2.wav';
%test = 'freqsaw/o100100.mp3';
test = 'misctests/sqr-100Hz.wav';
samples = 220500;

%Read in sample mp3 files
fn = sprintf( '../%s', test);
frm = fn(1,end-2:end);

if (frm == 'mp3')
```

$$test1, fs, NBITS, OPTS$$

```
= mp3read(fn, [1 samples]);
else
```

$$test1, fs, NBITS, OPTS$$

```
= wavread(fn, [1 samples]);
end

%Compute MFCC's for each file
```

```
testMFCC = melfcc(test1,fs,'wintime',w,'hoptime',v,'numcep',k);

%Eliminate lower 95% magnitude fractile coefficients
testThresh = quantile(testMFCC, thresh);

%Initialize thresholded copies of of the MFCC matrices
testInput = testMFCC;
sz = size(testInput);

for i=1:sz(1)
for j=1:sz(2)
if (testInput(i,j) < testThresh(j))
testInput(i,j) = 0;
end
end
end

%Perform ICA on test signal MFCC's
testOutI = fastICA(testInput, 'approach', 'symm', 'g', 'pow3', 'lastEig', 10,...
'numOfIC', 10, 'displayMode', 'off');

%Perform PCA on test signal MFCC's
```

$$testOutP, scores, latent, tsquared$$

```
= princompN(testInput);
testOutP = testOutP(:,1:12)';

% Commented out because most, if not all, test tones produce only 1
% independent component.
%ICA
% figure,SUBPLOT(3,3,1),scatter(testOutI(1,:),testOutI(2,:),3,'b'),xlabel('IC1'),ylabel('IC2');
% SUBPLOT(3,3,2),scatter(testOutI(1,:),testOutI(3,:),3,'b'),xlabel('IC1'),ylabel('IC3');
% SUBPLOT(3,3,3),scatter(testOutI(1,:),testOutI(4,:),3,'b'),xlabel('IC1'),ylabel('IC4');
% SUBPLOT(3,3,4),scatter(testOutI(1,:),testOutI(5,:),3,'b'),xlabel('IC1'),ylabel('IC5');
% SUBPLOT(3,3,5),scatter(testOutI(2,:),testOutI(3,:),3,'b'),xlabel('IC2'),ylabel('IC3');
% SUBPLOT(3,3,6),scatter(testOutI(2,:),testOutI(4,:),3,'b'),xlabel('IC2'),ylabel('IC4');
% SUBPLOT(3,3,7),scatter(testOutI(2,:),testOutI(5,:),3,'b'),xlabel('IC2'),ylabel('IC5');
% SUBPLOT(3,3,8),scatter(testOutI(3,:),testOutI(4,:),3,'b'),xlabel('IC3'),ylabel('IC4');
% SUBPLOT(3,3,9),scatter(testOutI(3,:),testOutI(5,:),3,'b'),xlabel('IC3'),ylabel('IC5');

% PCA
figure,SUBPLOT(3,3,1),scatter(testOutP(1,:),testOutP(2,:),3,'b'),xlabel('PC1'),ylabel('PC2');
SUBPLOT(3,3,2),scatter(testOutP(1,:),testOutP(3,:),3,'b'),xlabel('PC1'),ylabel('PC3');
SUBPLOT(3,3,3),scatter(testOutP(1,:),testOutP(4,:),3,'b'),xlabel('PC1'),ylabel('PC4');
SUBPLOT(3,3,4),scatter(testOutP(1,:),testOutP(5,:),3,'b'),xlabel('PC1'),ylabel('PC5');
SUBPLOT(3,3,5),scatter(testOutP(2,:),testOutP(3,:),3,'b'),xlabel('PC2'),ylabel('PC3');
SUBPLOT(3,3,6),scatter(testOutP(2,:),testOutP(4,:),3,'b'),xlabel('PC2'),ylabel('PC4');
SUBPLOT(3,3,7),scatter(testOutP(2,:),testOutP(5,:),3,'b'),xlabel('PC2'),ylabel('PC5');
SUBPLOT(3,3,8),scatter(testOutP(3,:),testOutP(4,:),3,'b'),xlabel('PC3'),ylabel('PC4');
SUBPLOT(3,3,9),scatter(testOutP(3,:),testOutP(5,:),3,'b'),xlabel('PC3'),ylabel('PC5');
```